WALKOWIAK

MICRO APPLICATION

5

AMSTRAD

LES JEUX D'AVENTURES ET COMMENT LES PROGRAMMER SUR CPC

UN LIVRE DATA BECKER

WALKOWIAK

MICRO APPLICATION

5

AMSTRAD

LES JEUX D'AVENTURES ET COMMENT LES PROGRAMMER SUR CPC

UN LIVRE DATA BECKER

Distribué par MICRO APPLICATION
147 Av. Paul Doumer
92500 RUEIL-MALMAISON

et également

EDITIONS RADIO
3 rue de l'Eperon
75006 PARIS

(c) Reproduction interdite sans l'autorisation de MICRO APPLICATION.

"Toute représentation ou reproduction, intégrale ou partielle, faite sans le consentement de MICRO APPLICATION est illicite (loi du 11 mars 1957, alinéa 1er de l'article 40).

Cette représentation ou reproduction illicite, par quelques procédés que ce soit, constituerait une contrefaçon sanctionnée par les articles 425 et suivants du Code Pénal.

La loi du 11 mars 1957 n'autorise, aux termes des alinéas 2 et 3 de l'article 41, que les copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à l'utilisation collective d'une part, et d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration".

ISBN 2-86899-018-5

Copyright (c) 1984 DATA BECKER
Merowingerstr. 30
4000 Düsseldorf
Allemagne de l'Ouest

Copyright (c) Traduction française 1985 MICRO APPLICATION
147 av. Paul Doumer
92500 RUEIL MALMAISON

Traduction Française et mise en pages assurées par Pascal HAUSMANN

Edité par Frédérique BEAUDONNET
Léo BRITAN
Philippe OLIVIER

TABLE DES MATIERES

PREFACE

1 INTRODUCTION 4
L'aventure aujourd'hui. Histoire et évolution.

2 LE CONCEPT 13
Ouverture. Fonctions. Réflexions préliminaires.

3 LA REALISATION 21
Organisation du monde. Digression: Variables et tableaux. Digression: READ DATA. Formatage de la sortie sur écran. Objets. Vocabulaire. Digression: Traitement des chaînes de caractères. Analyse de l'entrée. Aperçu schématique. Les conditions. Les actions. Programmation de l'exécution des instructions. Derniers pas. Listing.

4 PERFECTIONNEMENT 75
Save Game. Load Game. Digression: Stockage externe de données. Ergonomie. Motivation du joueur. Orientation et désorientation. Changement de lieu. Accès cachés. Limitations. Lumière disponible. Hasards. Du jeu d'aventure de texte au jeu d'aventure graphique.

5 LE SYSTEME DE JEU D'AVENTURE 116
Fichiers d'aventure. L'éditeur. L'interpréteur.

6 LA PRATIQUE DES JEUX D'AVENTURE 128
Règle du jeu. Conseils pour trouver la solution. Listing Château enchanté. Listing L'ivresse de l'or. Listing Space Mission. Listing Adventuregenerator. Listing Adventureeditor. Listing Adventureinterpreter. Listing Grafics Editor.

7 ANNEXE 249
Optimisation de la place en mémoire.

PREFACE

"JEUX D'AVENTURE (ADVENTURE GAMES), pour les uns le fin du fin, pour d'autres purement ennuyeux. Des jeux électroniques avec des textes importants et lors desquels le joueur est invité à participer à une série d'événements guidés par le hasard et à s'orienter à l'intérieur d'un labyrinthe."

Le lecteur intéressé trouvera cette définition dans l'annexe du manuel sur le CPC 464 et s'il devait lui accorder foi et écarter le thème des jeux d'aventure en s'exclamant "mon Dieu", il devrait renoncer à la façon la plus intéressante d'utiliser son ordinateur à des fins ludiques.

Et ce serait vraiment dommage car pourquoi devriez-vous renoncer à un plaisir goûté dans le monde entier par des millions de fanas de l'informatique ?

Car comment vous expliquerez-vous alors que vos camarades d'école ou vos collègues de travail passent leur récréation ou leur pause à discuter de la façon dont le héros peut renverser une situation critique à son avantage ?
Pourquoi trouve-t-on dans toute revue spécialisée des articles sur le thème de l'aventure ?
Et pour quelles raisons même des géants du domaine des loisirs tels que WARNER BROS ou LUCASFILM (noms protégés) pénètrent-ils sur le marché des jeux d'aventure et ne se contentent-ils pas des univers qu'ils créent et qu'ils représentent sur la pellicule ?

Pourquoi, si les jeux d'aventure ont vraiment si peu à offrir ?

Vous pourrez fournir une réponse à cette question pour vous-même et pour vos amis après avoir consulté ce livre.

Car avec le présent ouvrage vous tenez en main la clef ouvrant sur le monde des jeux d'aventure, un monde dans lequel devrait prénétrer tout être qui, par delà les choses sérieuses de la vie, n'a pas encore oublié l'engouement ressenti pendant les jours de son enfance passés à jouer.

Vous vivrez des aventures dans les temps les plus reculés et dans l'espace. Vous constaterez alors que les hasards dans ce livre sont le plus rarement laissés au hasard; que les labyrinthes ne sont jamais que

des accessoires destinés à mieux dérouter le joueur et que les jeux d'aventure ne doivent jamais et en aucune façon n'être que de simples jeux écrits, ceci étant d'autant plus vrai si vous utilisez un ordinateur tel que le CPC 464.

En me suivant vous développerez progressivement une conception sur la réalisation des jeux d'aventure en établissant leur représentation optique et la technique de jeu d'après des jeux professionnels.

Même si vous ne vous comptez pas parmi les pros de l'informatique vous découvrirez, grâce à de courtes digressions par la suite clairement signalées, les bases importantes pour la compréhension des programmes, de telle sorte que, même débutant, vous viendrez à bout de ce livre sans difficultés.

Vous serez aidé en cela par également la structure des programmes; après chaque démarche vous pourrez vérifier la justesse des reflexions dans leur application immédiate.

C'est ainsi que nous programmerons tout d'abord le monde d'action d'un jeu d'aventure, puis nous nous demanderons comment donner au joueur les moyens d'avancer et comment nous pourrons lui procurer des instruments avec lesquels il pourra travailler.

Un autre chapitre vous livrera les secrets particuliers de la programmation des jeux d'aventure; il vous fera des propositions supplémentaires qui vous permettront de compliquer la vie d'un participant.

En outre vous trouverez dans ce chapitre des instructions précises pour que vous puissiez construire un jeu d'aventure graphique à partir d'un jeu d'aventure de texte.

Et afin que la construction du graphisme ne demande pas trop de travail on vous donne aussi le listing d'un programmeur de graphisme qui produira pour vous les sous-programmes nécessaires.

Cependant on n'y trouve pas que ce programme auxiliaire mais avec le listing "Venturefix" nous vous procurons un générateur de jeux d'aventure qui vous permettra d'effectuer des programmes qui fonctionnent même si vous n'avez acquis votre CPC qu'aujourd'hui et ne disposez de ce fait d'aucune connaissance en programmation.

La chose est rendue encore plus facile par notre système de jeux d'aventure qui se compose de l'interpréteur qui en dépend et alors vous pouvez vérifier et corriger aussitôt chaque idée et chaque raisonnement pendant le jeu.

Si vous deviez toutefois préférer commencer tout de suite à jouer reportez-vousalors au dernier chapitre où vous trouverez les jeux d'aventure de texte 'L'ivresse de l'or' et 'Le château ensorcelé' ainsi qu'un listing pour le jeu 'Space mission'.

Je suppose que, plutôt que d'entendre un long discours, vous préférez vous intéresser au véritable sujet de ce livre et il ne me reste plus maintenant qu'à vous remercier chaleureusement pour l'achat de ce livre.

En outre je remercie tout particulièrement M. Achim Becker qui a permis la parution de ce livre ainsi que Mme Alicia Clees et M. Claus Wagner qui m'ont assisté pendant la rédaction sans compter leurs efforts.

Recklinghausen.
Décembre 1984

Jörg Walkowiak

CHAPITRE 1
- INTRODUCTION -

LES AVENTURES, ce sont des évènements extraordinaires dans la vie d'un individu qui sont caractérisés par des situations et des dangers extrêmes et sont gravés à jamais dans la mémoire de l'aventurier.
Elles exercent une grande influence sur le psychisme de l'homme et fascinent déjà de temps immémoriaux également des personnes qui n'y prennent pas part. La poésie utilisa cette particularité et à travers les époques se développèrent différents genres littéraires.
Ce furent tout d'abord les ménestrels qui narrèrent au peuple et aussi aux têtes couronnées les exploits de lointains héros. Puis, les poèmes des ménestrels se transformèrent au 12. siècle en épopées grandioses telles 'Lancelot' ou 'Perceval' de Wolfram von Eschenbach.
'Don Quichotte' et 'Münchhausen enthousiasmèrent également les foules.
Après cette époque où domina le roman picaresque, les récits de voyage (Defoe, Cooper, Karl May) reprirent le thème des aventures à leur compte jusqu'à ce que cette évolution trouve son aboutissement dans les romans policiers et de science fiction.

L'AVENTURE AUJOURD'HUI

Qui ne connaît pas les romans d'aventure cités dans les pages précédentes qui pourraient provenir d'un dictionnaire moderne; qui ne s'est pas laissé entraîner dans son enfance par les aventures de Old Shatterhands ou de Robinson Crusoë ? Qui ne s'est pas identifié avec le personnage principal et qui n'a pas imaginé son propre comportement dans une situation similaire ?

Il se peut pourtant que vous appartenez effectivement au petit nombre de gens qui pour des raisons de confort n'ont jamais lu un livre de ce genre. Pourquoi donc, demanderez-vous alors, si je peux me faciliter les choses !
J'ai vu Don Quichotte à la télévision, Karl May grâce aux directeurs de chaînes même plusieurs fois, pourquoi devrais-je passer mes loisirs à lire si je m'en arrange bien autrement. De toute façon, lire est dépassé depuis déjà bien longtemps.

Une chose est sûre, ce dernier argument serait censé vous donner raison, car si les chambres d'enfants contenaient autrefois des poupées, des boîtes de jeux de construction, des trains miniatures ou des livres pour enfants, ces derniers sont remplacés aujourd'hui par des engins télécommandés, des boîtes de jeux d'expériences, des magnétoscopes, des jeux vidéo et des ordinateurs. Dans ce domaine aussi, la technique est en progrès, ce qui est absolument à saluer, à la condition toutefois qu'on en fasse une juste utilisation.
On peut trouver en effet des utilisateurs d'ordinateur qui ne connaissent comme unique application que des batailles intersidérales et divers autres jeux de destruction ou d'action. Toutefois, ces utilisateurs constateront aussi un jour que le jeu se déroule toujours selon le même processus. Il n'est pas question de penser, seule la vitesse à laquelle on doit jouer en pressant les touches change un peu d'un jeu à l'autre.
Soumettons ainsi le marché des logiciels à un nouvel examen et l'on constatera que les possesseurs anglais et américains d'ordinateurs ont plus de choix; car outre les jeux d'Arcade déjà connus, les jeux d'aventure se sont taillé aux Etats-Unis et en Angleterre depuis longtemps une solide réputation.
Les jeux d'aventure représentent certainement avec les jeux de stratégie comme les échec ou Othello la forme d'utilisation d'un ordinateur la plus géniale si l'on s'en sert pour jouer.

Ces programmes permettent à tout utilisateur de vivre sans risque ni péril les aventures les plus intéressantes où même l'impossible peut devenir courant.
Imaginez un androïde que vous guidez avec de brèves instructions à travers tous les dangers, auquel vous donnez des instructions qu'il exécute ensuite à votre place et qui décident de la victoire ou de la défaite.
Ordonnez-lui d'aller d'aller dans une certaine direction et il vous informera alors de ce qu'il voit au nouvel endroit où il se trouve ou bien vous expliquera qu'il n'existe malheureusement aucun chemin dans la direction souhaitée.
Faites-lui examiner de plus près des choses qu'il trouve et ordonnez-lui, si vous l'estimez judicieux, de prendre ces objets.
Finalement vous croyez avoir trouvé pour chacun de ces objets et en fonction du but du jeu une possibilité sensée d'utilisation et faites agir l'androïde à cet effet; savourez votre triomphe ou bien faites-vous une raison de la rancune qui naît en vous à l'égard de ce jeu qui vous a captivé et détaché du monde quotidien.

Si vous avez chargé un jeu d'aventure dans la mémoire de votre ordinateur, vous ne disposez pas seulement pour votre jeu de quelques écrans multicolores mais vous vous mouvez dans un monde totalement décoré, un monde dont les mystères et les énigmes n'attendent que d'être découverts et résolus par vous.
Un bon jeu d'aventure vous fera agir ici comme dans la réalité, c'est-à-dire que action et réaction se trouvent en perpétuelle relation, les expériences liées à votre vie se révèlent utiles également dans les jeux électroniques.

Vous découvrirez que même un geste quotidien comme ouvrir une porte peut engendrer des difficultés qui s'annoncent par l'affichage d'une information telle que : "Je n'ai pas la bonne clef !"
Même la magie peut être nécessaire pour obtenir comme réponse "OK, la porte est ouverte", car seule l'imagination du programmeur fixe les frontières entre le possible et l'impossible.

Mais d'un autre côté un jeu électronique peut servir aussi comme méthode d'apprentissage. Par exemple un jeu muni de données sur notre système solaire serait ainsi plausible. Il nous permettrait d'étudier l'espace confortablement assis dans notre meilleur fauteuil.

Quelle méthode serait mieux appropriée pour apporter des connaissances

générales ou des informations spécifiques à des enfants ou des gens désireux de s'instruire, et ceci quel que soit leur âge ?

Peu importe si un jeu d'aventure a été conçu à l'origine comme programme d'enseignement ou seulement comme moyen de distraction, un bon potentiel de réflexes ne suffit plus, loin de là, pour gagner.
Une intelligence perçante et des qualités de logique intellectuelle vous seront absolument nécessaires pour venir à bout des nombreux dangers qui jalonnent la vie d'un aventurier:

Comment prendrez-vous du poisson afin de ne pas mourir de faim, si vous n'avez que vos mains nues pour pêcher ?
Comment pourrez-vous dissuader l'ours affamé de vous considérer comme son goûter ? Comment allez-vous faire pour traverser un cours d'eau impétueux de surcroît infesté de bêtes carnassières primitives ?

Tous ces problèmes peuvent être résolus avec un peu de réflexion, c'est à vous d'avoir seulement les bonnes idées et avec ça beaucoup d'imagination:

Votre tâche consiste à pénétrer dans une maison donnée, l'entrée étant pourvue d'un portail à claire-voie. En y regardant de plus près, vous apercevez par terre une clef quelques centimètres derrière la porte, mais malheureusement vos bras sont trop courts pour saisir la clef à travers la porte.

Comment parviendrez-vous dans la maison ?

Bon, passer par-dessus la porte est impossible, mais à quelques pas devant la maison vous avez remarqué des arbres. A un moment du jeu vous avez trouvé en outre un chewing-gum.

Ca a fait tilt ?

Nous avons eu aussi besoin d'un peu de temps pour la solution de ce problème, et ceci surtout parce que personne ne nous avait indiqué si précisément les objets utilisables:

Il n'y avait qu'à aller vers l'arbre et en casser une branche. Puis on revenait à l'entrée de la maison tout en mâchant bien soignement le chewing-gum, on fixait celui-ci à une extrêmité de la branche, on poussait le morceau de bois à travers les grilles de la porte jusqu'à la

clef qui restait collée ainsi à la branche; après avoir ramené prudemment le bout de bois on était enfin en possession de la clef.
Quelle chance que cette clef fût bien celle de la porte.

Une solution élégante, pas vraiment facile, mais déjà prévue comme cela pendant son élaboration par le programmeur. Je devrais ajouter il est vrai que la présentation de ce jeu spécifiait qu'il était nécessaire d'avoir déjà participé à des jeux.

L'exemple cité ci-dessus vous a peut-être transporté dans une 'atmosphère d'aventure', toutefois si vous ne deviez pas avoir remarqué le pouvoir de fascination qu'exerce ce qui est écrit, reportez-vous donc à la prochaine occasion à 'L'histoire sans fin'.

Quelles possibilités d'évolution découleront maintenant de l'introduction de bons romans dans les systèmes informatiques?

Il s'avèrera peut-être un jour nécessaire de compléter dans nos dictionnaires la définition du concept 'd'aventure':

Au 20. siècle, les progrès en matière de technique électronique permirent l'utilisation du roman d'aventure. Le précurseur en la matière fut Scott Adams avec le jeu 'Adventureland'; suivirent de nombreux autres jeux d'aventure, qui furent conçus, l'évolution aidant, comme de parfaites simulations du monde. De nombreux écrivains célèbres se rendirent compte en leur temps des nouvelles possibilités qui leur étaient offertes et créèrent à travers leurs oeuvres et pour beaucoup de gens un contre-poids au monde du quotidien.

HISTOIRE ET EVOLUTION DES JEUX D'AVENTURE

Après que l'on ait construit à partir de la première calculatrice un cerveau programmable, commençèrent dans les sphères des programmeurs les différents efforts pour inculquer aux ordinateurs la pensée humaine et le mode de comportement humain. C'est enfin en 1966 que Joseph Weizenbaum du Massachusetts Institute of Technology parvient à intéresser avec son programme ELIZA non seulement les spécialistes mais aussi le public.
Eliza, que l'on peut se procurer aujourd'hui pour tout type de micro-ordinateur dans des versions dépouillées, simule un psychiatre et imite sa technique caractéristique de conversation. Lors d'expériences avec des personnes-témoins, celles-ci se montrèrent après chaque séance très étonnées d'avoir fait part à une machine de leurs problèmes les plus intimes.

La continuelle évolution de l'intelligence artificielle apporta aux scientifiques un programme introduit sur un PDP-10: 'ADVENTURE - Colossal Cave'. Pour se changer les idées ces messieurs érudits purent mener des recherches dans un autre domaine, à savoir la chasse au trésor dans un monde obscur, dominé par le fantastique. Colossal Cave s'attira une très grande popularité dans les cercles de savants et, en 1979, fut finalement rendu également accessible à un vaste public. L'entreprise aujourd'hui bien connue de tous MICROSOFT publia 'Microsoft's Adventure', une version sur disquette de ce premier jeu destinée au micro-ordinateur le plus en vogue aux Etts-Unis à cette époque, le TRS-80.

Toutefois, c'est dès 1978 que les jeux d'aventure furent rendus praticables à la maison et ce grâce à un jeune Américain du nom de Scott Adam, lequel avec son ADVENTURELAND posa la première pierre du nouveau-venu ADVENTURE INTERNATIONAL.
Dans ce jeu il s'agit également de trouver différents trésors dans un monde mystique, ce qui est rendu naturellement difficile par des dragons, des labyrinthes, des courants de lave en fusion et autres surprises. Tout commence de façon très anodine au milieu d'un bois. Pourtant, lorsque l'on s'est enfin situé et que l'on sait quelle direction on doit prendre et qu'après coup on trouve l'entrée menant à l'empire des morts, c'est alors que...

ADVENTURELAND fit l'objet d'une demande gigantesque et pour la bonne raison qu'il n'y avait pas encore des jeux d'une qualité égale à celle des jeux d'arcade. Bientôt, en raison de cet énorme succès, on tira de

cet ADVENTURELAND une série de douze jeux d'aventure, où l'intérêt de la clientèle était maintenu grâce à un choix judicieux des sujets.
Si vous avez toujours rêvé d'en découdre avec le comte Dracula, dans 'The count' vous pouvez le faire. Ou alors si vous préférez partir à la recherche d'un saboteur à l'intérieur d'une centrale atomique, pas de problème, 'Mission impossible' rend cela possible.

A cette époque et profitant du même courant, un type de programme portant le titre de'INTERACTIVE FICTION', apparut comme concurrent sur le marché américain du logiciel. Ces programmes devaient représenter un nouveau genre de littérature, devaient, comme le nom l'indique, être des livres incitant à participer. Tout d'abord le joueur n'est qu'un simple lecteur, il est introduit après le début du programme de façon trés précise dans l'action. De nombreuses pages scintillent sur l'écran, elles donnent au lecteur des renseignements sur les événements passés, la situation actuelle, sur ce qu'on ne voit pas etc, et elles se lisent exactement comme un livre.

Ainsi 'Local Call For Death' commence comme une histoire policière: trois hommes se rencontrent à leur club, au cours de la soirée l'un d'entre eux est appelé au téléphone. C'est le neveu qui appelle son oncle, visiblement énervé, pour lui demander de l'argent. L'oncle refuse car il sait bien par expérience que cet argent serait perdu aux courses. Le lendemain matin on lui annonce que son neveu s'est suicidé.

C'est à ce moment qu'intervient alors le joueur. Il joue à partir de maintenant le rôle d'un des personnages principaux et essaie en discutant directement avec les participants d'élucider le soi-disant suicide.

La caractéristique de ces programmes est que l'on se fixe clairement un but unique, comme ici l'explication d'un meurtre.
Pour atteindre ce but, nous devons en tant que joueurs, nous servir moins de quelque objet que de questions judicieuses. Qu'est-ce qui est le plus intéressant: lire une conversation dans un livre ? Suivre un interrogatoire dans un film ? Ou bien vivre une expérience couronnée de succès telle que confondre un criminel grâce à ses propres facultés de raisonnement ?

C'est justement cette possibilité de dialoguer qui fait le charme de 'Interactive Fiction'. Un autre jeu à citer en exemple: 'Encounter in the Park', paru également chez Adventure International.
Dans ce jeu nous rencontrons pendant la promenade du matin la femme de

nos rêves et notre but avoué consistera justement à séduire cette jeune fille et à l'amener à nous donner son consentement. A cet effet, le programmeur a mis à la disposition du joueur tout un éventail de techniques de persuasion et on s'amuse vraiment beaucoup quand on voit la façon dont réagit cette dame à certaines propositions.
Le pire qui puisse nous arriver si on lui fait des propositions vraiment peu sérieuses est que, indignée, elle se détourne de nous et nous sommes informés que par dépit nous avons passé le reste de notre vie dans un cloître.

Comme déjà mentionné, l'intérêt porté pour toutes ces aventures de texte était très grand, des entreprises toujours plus nombreuses publièrent en conséquence de plus ou moins bonnes copies de ces jeux. Stimulés par la pression croissante de la concurrence, quelques producteurs de logiciels imaginèrent de nouvelles générations de jeux d'aventure.
Parurent alors les premiers LABYRINTH-ADVENTURES, qui, en fonction de leur nature, mettaient le joueur dans l'obligation de s'échapper vivant d'une bâtisse. Presque toujours un jeu finissait de telle sorte que le joueur se perdait sans espoir de trouver une issue dans le labyrinthe représenté en perspective à l'écran.

La phase suivante de l'évolution fut les QUEST-ADVENTURES, qui par la suite furent programmés le plus souvent également comme des Real-Time Adventures. Avec ce rejeton de la grande famille des jeux d'aventure, le joueur est soumis à un effet de stress supplémentaire car si, lorsqu'apparaissent quelques brigands ou autres manants, il n'appuie pas aussitôt sur F pour Fight et qu'après cela en cours de partie il presse n'importent quelles touches ou même sciemment pour brandir son épée dans une direction bien précise, le jeu finissait prématurément en raison de trop grandes blessures reçues par le personnae principal.
Malheureusement il découle de la conception de ces jeux une importante limitation de la marge d'action du joueur, car il ne reste à celui-ci qu'un échantillon pauvre de quelques actions comme combattre, négocier, acheter ou fuir, qui lui sont offertes au menu, d'où le nom de QUESTion.

La phase suivante intervint grâce à la chute des prix des processeurs de graphisme et des éléments de RAM, car à présent les constructeurs d'ordinateurs n'avaient plus de difficultés pour procurer à l'utilisateur privé un graphisme de bonne qualité avec des ordinateurs à des prix abordables, ce qui autrefois n'était réalisable que sur des appareils bien plus grands et de ce fait également plus chers.

Ainsi commença en 1982 (pour les utilisateurs d'Apple un peu plus tôt) avec la commercialisation du premier GRAFIKADVENTURES un nouvel essor pour ce type de jeux. Car s'ils avaient jusqu'ici été mis à l'index par de nombreux possesseurs d'ordinateur comme de purs jeux à texte, ils devenaient intéressants du jour au lendemain en raison de leurs nombreuses images multicolores; en outre ils se révélaient parfaitement appropriés à arracher de la bouche des membres de la famille incultes en matière d'informatique et qui demandent: "A quoi sert donc ton ordinateur ?", au moins des paroles teintées d'étonnement:"Comme c'est joli."

La valeur du jeu et le but ne s'étaient absolument pas modifiés. Mais soudain de nouvelles couches d'acheteurs étaient accessibles; une raison pour jeter sur le marché de nombreux jeux connus depuis des années, dans leur nouvelle présentation: les descriptions exhaustives avaient disparu, à la place on pouvait voir sur le petit écran les données de la situation du moment.

On hésite encore à dire que les jeux étaient mieux utilisables ou plus faciles à jouer, pour ma part je préfère être informé par une annonce du genre:

"Je suis dans la forêt. Autour de moi je ne vois rien que des arbres. Entre deux chênes je vois un trou dans la terre".

que de voir quelques arbres; je considère ceux-ci prématurément comme une partie de forêt tout à fait normale, c'est pourquoi je passe rapidement sans examiner les arbres outre mesure et omets de ce fait le trou dans la terre, une petite tache noire sur le moniteur. Bien entendu c'est précisément là que repose une partie du trésor ou bien un objet sans lequel il est impossible de réussir.

CHAPITRE 2

- LE CONCEPT -

Dans les chapitres suivants nous développeront ensemble des jeux d'aventure. Naturellement, nous souhaitons pour notre oeuvre un profil très professionnel, c'est pourquoi nous n'hésiterons pas à analyser les qualités et les caractéristiques spécifiques des programmes que l'on trouve sur le marché.
Nous enchaînerons en nous demandant comment nous programmerons les différentes fonctions en BASIC.
Nous voulons attirer ici l'attention sur le fait que ces diverses routines sont construites de telle façon qu'elles peuvent être utilisées non seulement pour un jeu précis mais aussi et sans modifications dans tous nos programmes d'aventures.

L'OUVERTURE

Il a déjà été établi qu'un jeu d'aventure a pour but de transporter le joueur dans un autre monde, un monde qui rend l'impossible possible. Ce joueur doit y vivre des aventures, doit donc pouvoir se mouvoir et agir, il doit être informé du résultat de ses efforts afin de pouvoir entreprendre d'autres actions.
Une action sensée visant à un but précis n'est possible pour l'individu qu'après toutefois avoir fait usage de ses sens. Les informations que reçoit l'aventurier grâce à ses yeux, ses oreilles et son sens du toucher, doivent être offertes de ce fait au joueur sous une forme correcte.
Etant donné que, avec les connaissances actuelles, le cerveau humain ne peut être raccordé directement à un ordinateur, des réponses aux questions: "où suis-je? Que vois-je ? Où puis-je aller? Qu'est-ce que je ressens ?" doivent nous être données sous une autre forme, ce qui se fait ici sous la forme de phrases courtes mais complètes.

Ce n'est que dans les cas les plus rares que la perception du monde environnant se fera de façon consciente. Le contenu d'un espace est enregistré, sauf demande particulière, en quelques secondes; c'est pourquoi on ne peut aussi demander au joueur d'étudier en premier lieu des dissertations volumineuses sur le lieu dans lequel il vient de

pénétrer.
Même s'il ne passait que cinq à dix secondes à lire plusieurs lignes de descriptions, la concentration qu'il y consacrerait bloquerait le cours du jeu et empêcherait le joueur de s'identifier avec le personnage principal et de se laisser pénétrer par l'illusion d'un monde imaginaire.

Pour contourner ce problème, on limite la sortie du texte à un minimum d'informations acceptable et on présente celles-ci, dans la mesure où il ne s'agit pas de jeux graphiques, en deux zones sommaires sur le moniteur et qui ont chacune une fonction bien définie.

Ainsi, la moitié supérieure de l'image nous indique où nous nous trouvons à l'instant et ce que nous voyons. Par contre, la moitié inférieure est destinée à la communication avec le personnage central qui agit pour nous. Ici nous entrons nos instructions et nous y apprenons ce qui résulte de nos entrées.

Un exemple:

"Je suis dans une forêt sombre.
Je vois beaucoup de vieux arbres.

Je peux aller vers le nord, l'est, l'ouest.

"Que dois-je faire ?"

Dans le cas où nous envisageons d'avancer dans le jeu le plus rapidement possible, il serait faux de progresser sans but dans l'une des trois directions.
La première chose qu'un participant à un jeu d'aventure devrait faire dans toute situation nouvelle, est de tout analyser avec minutie. On pourrait obtenir comme réponse à la question: "Examine arbres", quelque chose comme: "Il s'agit de vieux chênes." ou "Même ici la forêt meurt".
Certes, ces réponses ne nous éclairent pas beacoup, elles nous montrent pourtant qu'apparemment il n'est pas si utile de tout examiner, mais comment pourrait-on connaître la réponse sinon en posant la question. Car une réponse comme "Je vois une ouverture dans un tronc d'arbre" aurait

été possible. Il se peut qu'il s'agisse d'un vieux nid de pivert dans lequel se trouve une partie du trésor que l'on cherche; "Examine ouverture" nous le dira.
Si nous n'avions trouvé aucun point de repère, même après la question "Examine forêt" (il pourrait s'agir aussi d'une forêt enchantée), nous devrions enfin décider de la direction à prendre alors, guidés par la raison.
Comment vous comporteriez-vous dans une telle situation, sans carte ni boussole, seul et sans aide, éloigné de tout chemin ?
Vous monteriez peut-être dans un arbre pour voir ce qu'il y a à proximité et au loin.

Votre entrée devrait être ainsi formulée: "Grimpe arbre", et si vous ne recevez pas une information du genre: "Je ne suis pas assez sportif pour ça!", l'image du moniteur se modifiera:

"Je suis dans la ramure d'un vieux chêne.
Je vois des montagnes au nord,
à l'est un lac,
à l'ouest s'élève de la fumée.

Je peux aller vers en bas

GRIMPE ARBRE
D'accord !

Que dois-je faire ?"

Comme cela toute l'affaire se présente beaucoup mieux, car après l'entrée de B pour en Bas, s'offre à nous le problème de décider quelle région nous allons explorer en premier.

L'information "D'accord" nous indique que l'entrée a été comprise et exécutée. Si cela n'avait pas été le cas, nous aurions obtenu par exemple: "Je ne comprends pas le verbe !", ce qui nous amènerait certainement à rechercher une autre forme d'expression convenant au programme. Ceci est bien entendu également valable pour l'objet de notre entrée.

On peut bien sûr envisager deux autres réponses. Pendant le cours d'un jeu, l'une nous indiquera souvent: "I must be stupid, but ... -Je ne comprends pas ce que tu veux dire !", par laquelle le programme nous indique qu'il considère notre entrée comme n'ayant aucun sens, ce qui dans le cas de: "Tue arbre" serait absolument fondé. En tout cas, cette action n'est pas prévue alors dans le jeu.

Il en va de même avec l'annonce standard "You can't do that ... yet. - Cela ne va pas pour l'instant." qui nous signale que les mots utilisés appartiennent bien au vocabulaire programmé, mais que l'ordre ne tombe pas à un bon moment: "Prends clef" sera compris par le programme du jeu d'aventure si une clef intervient dans le jeu, mais n'est pas correct aussi longtemps que cette cette clef ne se trouve pas dans le même espace que le joueur.

Pourtant nous pouvons respirer, car si nous devions obtenir cette réponse, nous sommes sur la bonne voie; toutes les conditions necéssaires ne sont cependant pas encore remplies pour exécuter l'ordre. Ainsi une porte verrouillée pourra être ouverte si nous nous sommes procuré un instrument approprié. Si la réponse à: "Ouvre porte " était: "Je ne comprends pas ce que tu veux dire.", alors les efforts suivants resteraient sans effet.

REFLEXIONS PREALABLES

La structure générale des jeux d'aventure nous est maintenant connue. Nous avons une idée de la façon dont notre jeu doit se présenter sur le moniteur et quelles informations doivent être préparées. Dans ce qui suit nous nous intéresserons de plus près aux différents éléments de notre système de jeu d'aventure et nous élaborerons une construction qui sera réalisée dans le prochain chapitre.

Les espaces dans le jeu d'aventure:

Le monde du jeu d'aventure se compose des lieux du jeu correspondant, c'est-à-dire que chaque endroit accessible au joueur embarqué dans une aventure est appelé lieu. La grandeur de ce dernier n'a vraiment aucune importance, cela peut être comme dans nos exemples précédents une forêt, la ramure d'un arbre, il peut s'agir aussi d'une penderie ou de l'intérieur d'une voiture.
Les différents lieux se distinguent pour le joueur selon leur description et leur disposition géographique; d'un point de vue de technique de programmation, ils se distinguent, comme nous le verrons plus tard, également par leur numéro de lieu.

Le joueur peut atteindre tous ces lieux qui doivent alors être reliés entre eux de façon correcte. Dans chaque jeu, il est possible d'aller dans les quatre directions correspondant aux quatre points cardinaux: nord, sud, ouest et est. Certains permettent même un mouvement vertical.
Naturellement il ne peut se produire que le joueur quitte le lieu 5 en direction de l'ouest, pénètre dans le lieu 6 par l'est, et lorsqu'il veut retourner au lieu 5, ne trouve aucun chemin vers l'est, ou pire, que par un mouvement vers le bas il parvienne à nouveau au lieu 5.
Pour prévenir les critiques d'aventuriers chevronnés, il faut dire qu'ici aussi des exceptions confirment la règle, car que serait un jeu d'aventure sans un labyrinthe magique.

Les objets

A chaque lieu correspondent en outre les objets qui lui appartiennent de façon caractéristique. Nous devons placer tous les objets dans les lieux ou, plus correctement, attribuer un lieu à chaque objet. Cette attribution nous permet d'éviter que des objets existent en double.

Un autre problème se pose à nous avec le rangement des objets qui ne peuvent être présents au début du jeu. Où restent-ils jusqu'à leur apparition ?

Supposons que notre héros entre dans un lieu et que n'apparaisse comme objet visible qu'une vieille caisse en bois fermée à clef.
A l'entrée effectuée par le joueur "Ouvre caisse", les règles de la logique exigent que la caisse disparaisse de l'écran et qu'à la place apparaisse une vieille caisse ouverte remplie si possible de pièces d'argent.

Eh bien, la solution de ce problème ne nous créera pas de trop grandes difficultés. Dans notre jeu, nous mettrons en place un lieu supplémentaire, l'entrepôt, où nous stockerons tous les objets non encore utilisés à ce moment. Si nous ne programmons aucune voie de communication avec ce lieu, le joueur n'y aura pas accès et ne pourra ainsi, contrairement à ses souhaits, saisir ces objets.

Dans le chapitre suivant, nous verrons comme il est facile de programmer l'entrepôt et même le monde du jeu d'aventure dans son ensemble; dans la suite de ce livre, nous nous rendrons compte aussi du caractère indispensable d'autres lieux particuliers.

Les entrées

Comme il ressort du paragraphe précédent, les entrées du joueur se composent le plus souvent d'un verbe et d'un complément d'objet. C'est pourquoi notre programme devra tout d'abord contrôler si une entrée est permise et pour cela une séparation entre verbe et complément sera obligatoire. Si le joueur commet une faute d'orthographe ou si le verbe utilisé n'est pas prévu, un message approprié doit être donné. Si notre jeu retrouve le verbe de l'entrée dans son vocabulaire, le complément doit être vérifié de la même manière.
Si les deux mots sont définis, le programme peut réagir de la façon prescrite.

Lors de l'élaboration de ce vocabulaire, le programmeur doit du reste agir avec une extrême minutie car il doit prévoir toutes les entrées possibles que n'importe quel joueur pourrait faire.
Un de mes amis qui n'avait jamais joué auparavant à un jeu d'aventure me prouva dernièrement que cette entreprise peut se révéler ô combien ardue.

En effet, après avoir étudié l'en-tête et les instructions nécessaires, il voulut résoudre d'un coup tous les problèmes avec "Trouve trésor", une entrée à laquelle en tant que programmeur je n'avais naturellement pas pensé.

En outre l'emploi de synonymes deviendra nécessaire, car si le joueur devait perdre trop de temps à exposer ses souhaits au jeu, il perdrait vite tout intérêt pour le jeu.

En ce qui nous concerne nous Français, le choix des mots peut vraiment devenir problématique, car là où l'Américain peut se faire comprendre aisément avec "Climb Tree", "Shoot Gun", "Drop Box", "Look", il n'en va pas vraiment de même pour nous avec "Grimpe Arbre", "Tire Fusil", "Pose Caisse", "Regarde".

Informations

La première sortie apparaissant sur l'écran qui ne peut plus compter comme description du lieu, indique au joueur les points cardinaux en direction desquels il peut quitter l'endroit où il se trouve.

En outre, l'exécution d'une action précise comme aussi d'une éventuelle réaction, doivent être montrées au joueur de façon très nette. Dans le cas le plus simple, ceci s'effectue avec "O.K.", la plupart du temps des informations supplémentaires sont toutefois sorties. Parallèlement, les entrées les plus diverses appelleront souvent la même réponse, des réponses qui justement pour cette raison durent être enregistrées et standardisées. Par exemple "Je ne suis pas si fort", "Cela n'a vraiment aucun sens!" et "Je ne comprends pas ce que tu veux dire!" apparaîtront fréquemment.

Fonctions standard des jeux d'aventure

Dans un jeu d'aventure moyen on trouve 30, 50 lieux différents, dans chaque lieu un ou plusieurs objets. Ces objets peuvent eux-même contenir des choses de toute sorte, et comme dans un jeu d'aventure on ne sait jamais exactement quel usage peut être fait plus tard de quelques trucs, beaucoup de joueurs ont tendance à emporter tout ce qui leur tombe dans les mains et qui n'est pas trop lourd à porter.
Garder une vue d'ensemble devient alors bien difficile, car étant donné

que nous en prenons tous à notre aise, nous n'avons naturellement pas pris de notes.
Par chance, les programmeurs s'en sont rapidement aperçu, de sorte que chaque jeu d'aventure indique aujourd'hui sur l'écran après l'entrée de INVENTAIRE tous les objets que nous portons sur nous.

Dans le cas d'un jeu d'aventure assez simple ou conçu pour débutants, nous pouvons, dans des situations délicates, nous faire donner des tuyaux. L'ordre HELP - AU SECOURS doit être alors introduit.

A l'inverse des entrées standard verbe et complément, il s'agit ici, comme du reste aussi avec l'inventaire, de ce qu'on appelle un ordre à un mot, lequel dans ce cas n'introduit toutefois pas une action.
Pourtant, ce mot peut apporter beaucoup de choses:

"J'examinerais tout" nous montre notre négligence. "Une fine silhouette apparaît et écrit dans le sable la phrase "Sésame, ouvre-toi!" nous fera par contre bondir de joie dans une situation délicate.
Cela va déjà moins bien lorsque nous devons payer cette aide: "Un lutin apparaît et dit que 10 points nous seront retirés pour son aide. Il veut savoir si nous avons besoin de son aide."
De sombres nuages apparaîtront probablement à l'horizon si nous obtenons pour notre demande d'aide la réponse suivante: "'Au secours' n'est pas prévu dans ce jeu !" ou si réapparaissent pour toute réponse les règles générales du jeu.

Il peut assurément être énervant de ne pouvoir se sortir d'une certaine situation, pourtant personne ne devrait considérer les deux derniers exemples comme un acte de méchanceté de la part du programmeur; si la fonction d'assistance était assurée de bout en bout, certains petit malins gagneraient en 30 minutes, mais ne pourraient jamais se rendre compte combien un tel programme peut être fascinant.
C'est pourquoi certains programmes font attention à l'utilisation de cette fonction et finalement refusent toute aide ou induisent même le joueur en erreur en cas d'usage trop fréquent.

CHAPITRE 3

- LA REALISATION -

INDICATIONS SUR LES LISTINGS

Avant de nous attaquer à la programmation, je voudrais vous donner encore quelques indications de nature plutôt technique et qui doivent servir à ce que le plaisir que vous éprouvez à travailler avec ce livre ne soit pas gâché par une erreur éventuelle.

Les lignes de programme suivantes ne seront sûrement pas les premières que vous entrez avec peine par l'intermédiaire du clavier, et ainsi vous aurez également fait l'expérience que des fautes de frappe ne cessent de se glisser malgré toutes précautions.

Prédestinées pour ce genre de fautes sont les lignes DATA, surtout lorsqu'elles ne contiennent que des chiffres comme ce sera ici aussi le cas.
De plus, il est indispensable d'entrer certaines lignes exactement comme elles sont écrites. Cela vaut en particulier pour le nombre d'espaces dans les chaînes de caractères.

Pour vous donner un moyen de contrôle efficace, tous les listings de ce livre sont reproduits de telle façon que vous devez les voir apparaître à l'écran après avoir entré LIST.

De ce fait, si vous avez l'impression que quelque chose ne fonctionne pas comme prévu, ne vérifiez tout d'abord que les caractères figurant à l'extrême droite d'une ligne. Si elles ne concordent pas, en toute probabilité, vous avez déjà trouvé la ligne de programme incorrecte.

En outre vous ne devriez utiliser que les numéros de ligne indiqués, ce qui garantit un bon fonctionnement de votre programme.
Car dans le développement suivant, quelques lignes de programme seront remplacées et des routines supplémentaires seront également insérées.

L'IDEE DU JEU

Ainsi, nous sommes venus à bout d'un minimum de théorie ennuyeuse, tout ce qui nous manque est une histoire que nous allons appliquer sous la

forme d'un jeu d'aventure. Nous pouvons alors brancher notre CPC 464 et aborder la pratique. Vous avez peut-être déjà une idée précise de votre premier jeu d'aventure. Si ce n'est pas le cas, inspirez-vous donc des propositions suivantes.

1. LA MAISON HANTEE
Vous recevez la dernière lettre de votre tante qui vous apprend qu'elle vous lègue une vieille maison très luxueuse. Vous êtes toutefois surpris, car elle vous indique que vous devez vous comporter avec prudence car il pourrait sinon vous arriver ce qui est arrivé à votre oncle. De tels propos ne vous empêchent pas de visiter la maison. Tout de suite vous constatez toute une série de choses bizarres, vous trouvez une bibliothèque secrète avec des livres sur l'exorcisme, vous découvrez dans une cave un autel de sacrifice. De nombreux esprits vous rendent la vie dure jusqu'à ce que vous découvriez finalement que l'un de vos ancêtres a tué jadis le chef d'une secte pour établir son propre pouvoir et qu'en guise de punition, il a été emmuré vivant. Vous devez maintenant l'aider à trouver le repos éternel.

2. LE TESTAMENT DU COPAIN
Votre ami et voisin, un savant renommé, est mort dans des circonstances mystérieuses. Vous recevez un papier dans lequel il vous demande d'achever son oeuvre. Après avoir réussi à pénétrer dans son laboratoire jusque là secret, votre tâche consiste à faire fonctionner son superordinateur. Celui-ci vous donne par la suite des indications supplémentaires.

3. STAR ODYSSEY
Votre vaisseau spatial est si endommagé lors d'une étrange tempête que vous devez atterrir sur une station déserte pour vous procurer des pièces de rechange afin de pouvoir retourner vers la Terre.

4. L'IVRESSE DE L'OR
Vous rencontrez dans le désert un homme blessé à mort. Celui-ci vous livre des renseignements sur sa mine d'or. Il vous montre quelques pépites et vous raconte qu'il aurait caché de l'or à sept endroits différents sur son exploitation; comme il n'en a plus besoin, il vous suggère d'aller le chercher.

Je souhaiterais prendre cette dernière proposition, un jeu du type chercheur d'or, comme base pour les prochains chapitres de cet ouvrage et

l'élaborer avec vous. Bien entendu vous êtes libre d'introduire vos propres idées, toutefois, lors de toute extension encore à présenter, je me rapporterai toujours à "L'ivresse de l'or".

L'ASPECT DU MONDE

Avec le choix du thème s'établit parallèlement le déroulement grossier du jeu. Dans ce qui suit nous allons élaborer et parfaire continuellement différentes structures jusqu'à ce qu'un monde soit conçu jusque dans ses moindres détails.

Lors de la programmation, nous procèderons exactement de la même manière; nous structurerons notre jeu progressivement et nous nous assurerons que les fonctions attribuées aux différentes routines sont correctes.

LA CARTE

Au commencement il y a le néant. Donc nous créerons dans ce qui suit ce monde nécessaire de façon si urgente à notre jeu d'après nos idées. Ce que doit offrir notre monde, le thème en décide. L'ivresse de l'or se déroulera probablement dans et autour d'une exploitation minière.

Imaginons un peu la mine: un versant de colline pierreux; une cabane à outils; des rigoles en bois qui amènent l'eau d'un ruisseau nécessaire au lavage de l'or; des poutres vermoulues, sèches, qui bordent la sombre entrée; d'obscures galeries parsemées de dangers.

A coup sûr, de nombreuses images de cette sorte nous viendrons encore sans grandes difficultés à l'esprit, des images qui nous apporterons une quantité suffisante de lieux pour l'action prévue.

Je voudrais limiter notre exemple concret pour commencer avec six lieux. Nous commencerons par une ballade dans une forêt qui nous conduira jusqu'à la mine où motre tâche consistera à chercher les trésors.

De cette façon, nous avons mis en place le début provisoire de l'ivresse de l'or. Pour pouvoir toutefois décider de la victoire ou de la défaite du joueur, nous devons également établir une fin de jeu couronnée de succès, en progressant dans notre projet nous devons prévoir aussi des situations menant à l'échec.

Nous pouvons ainsi nous contenter de trouver tous les trésors, nous pouvons donner à l'aventurier pour tâche de déposer le trésor dans un endroit sûr. Pour commencer, la découverte de la cachette de l'or sera pourtant suffisante.

Tâche et but sont définis ainsi que deux endroits d'action essentiels: la forêt, d'où nous partons et que nous structurerons avec deux lieux, et la mine. De plus il est clair que nous ne pouvons laisser se dérouler aucune action dans ce petit monde. Car nous avons besoin de place pour pouvoir cacher quelques objets et tendre des pièges au joueur.

Normalement une exploitation minière n'a pas son entrée entre les racines d'un arbre, nous devons créer des lieux de transition dans le paysage pour configurer le jeu de façon réaliste.

Trois autres lieux nous procurent suffisamment de place pour élaborer la première version de notre jeu, nous pouvons maintenant dresser une liste de tous les endroits qui apparaissent:

dans la forêt
dans la forêt
une forêt arrêtée par une paroi rocheuse
une clairière
clairière contre la paroi rocheuse
entrée dans la mine

La démarche suivante pour fabriquer le programme est logiquement de dresser une carte reproduisant tous les endroits sous forme de rectangles. Ces rectangles sont repertoriés et munis de la description du lieu. Nous faisons alors attention à une formulation correcte et nous tenons compte si possible d'une longueur de ligne de 40 caractères que nous utilisons sur notre CPC 464. De la sorte, notre description doit s'apposer sans problème à l'introduction "Je suis".

Le numéro que reçoit chaque lieu ne joue aucun rôle. Nous devons seulement commencer par un et suivre l'ordre logique, ce qui nous évite une double numérotation. Après quoi nous relions les lieux entre eux et notons à chaque limite la direction correspondante.

Nous pouvons maintenant voir d'après cette carte que, si le joueur se trouve par exemple dans le lieu 3, les informations suivantes doivent apparaître sur le moniteur:

"Je suis dans la forêt devant une paroi rocheuse."

Notre carte nous donne en plus une indication sur les directions où il nous est possible d'avancer :

"Je peux aller vers l'ouest, le sud."

Dans notre programme nous pourrions introduire alors pour chaque lieu une condition du genre "Si joueur en lieu 3, imprime 'Je suis dans la forêt devant une paroi rocheuse.'". Mais à ce rythme nous aurions rempli notre mémoire très rapidement.

Le principe est toutefois correct. Nous mémoriserons tous les lieux dans un tableau de variables et nous accèderons ensuite chaque fois à l'élément dont nous avons besoin pour sortir la description d'un lieu.

DIGRESSION : VARIABLES & TABLEAUX

Pour mémoriser des résultats intermédiaires ou des données d'un programme, on utilise dans chaque langage informatique des variables.
Ces variables se comportent comme des petites cases dans lesquelles on peut déposer quelque chose pour y être conservé. Ces cases portent des noms, 'coffre' ou 'tiroir' par exemple. On peut trouver aussi des cases de même nature: dans le cas du tiroir, tiroir 1, tiroir 2, 3, etc. Chaque case s'appelle tiroir et ne se distingue des autres que par son numéro. C'est pareil en Basic: nous pouvons donner à chaque variable un nom spécifique ou munir des variables de même nature d'un nom unique et d'un numéro d'index, ce dernier sera alors écrit entre parenthèses derrière le nom de la variable.
On distingue alors deux types fondamentaux de variables. Les unes ne portent que des chiffres avec lesquels on peut effectuer des calculs; les autres variables enregistrent les textes. La composition et la longueur des textes ne jouent aucun rôle, ou plus précisément: nous permettrons

l'utilisation de chaque caractère de notre clavier et des longueurs allant jusqu'à 255 caractères consécutifs.
Exemples corrects : A, A1, A2, TEXTE1 pour des variables numériques (que des chiffres), et A$, A1$, A2$, TEXTE1$ pour des variables de chaînes de caractères. Par rapport à ces variables simples, les variables indexées correspondantes seraient A(1), A(2), A(3) ou A$(1), A$(2), A$(3).
En outre le CPC 464 vous permet une distinction des variables numériques en chiffres entiers et en nombres à virgule flottante.
Pour matérialiser le concept de variables, pensez aux casses (ensemble de cases prévues pour conserver les lettres en plomb des typographes). Imaginez qu'une variable soit une case d'une casse.
Poursuivons notre comparaison:
De notre carte de jeu d'aventure, enlevons les rectangles des différents lieux. Déposons-les chacun dans une case, faisons cependant attention à ce que la disposition préalable des lieux reste la même, à savoir tout en haut à gauche se trouve notre forêt, dans la deuxième rangée à l'extrême gauche se trouve la première galerie.
A la question où se trouve la mine, nous pourrions répondre par 'dans la première colonne de la deuxième ligne', de la même façon, les positions de tous les autres lieux sont fixées par les coordonnées de ligne et de colonne. Cette disposition est appelée en langage informatique tableau bidimensionnel, ou, ce qui est plus joli, Array.
En Basic nous appellerions notre casse tableau de variables CARTE$ ($ parce que le texte doit être mémorisé), et nous attribuerions aux différentes variables le contenu suivant:
CARTE$(1,1) = "dans la forêt" première ligne, première colonne
CARTE$(1,3) = "a la paroi rocheuse" première ligne, troisième colonne
CARTE$(2,1) = "entrée dans la mine" etc.
A partir de la ligne 500 nous entrons donc les lignes suivantes:
```
500 lieu$(1)="dans la forêt."
510 lieu$(2)="dans la forêt,"
520 lieu$(3)="dans la forêt devant une paroi rocheuse."
530 lieu$(4)="dans une clairière."
540 lieu$(5)="dans une clairière."
550 lieu$(6)="devant la galerie d'une mine."
```
Ainsi, les lieux existent dans notre ordinateur, il nous manque encore le personnage du jeu qui avance dans le monde programmé et nous décrit la situation dans laquelle il se trouve à chaque instant. Le lieu où il se trouve dépendra naturellement de nos entrées; nous introduisons à cet effet une autre variable que nous appelons JOUEUR qui mémorise la situation du joueur. Les bonnes valeurs varient jusqu'ici pour ces variables entre 1 et 6, car PRINT LIEU$(JOUEUR) fait apparaître alors la

description correspondant à l'endroit où se trouve le joueur. Faisons un essai en entrant les lignes suivantes:
Attention! Veillez à ne pas modifier les numéros de ligne!

```
1140 PRINT"Je suis";
1150 PRINT lieu$(Joueur)
1390 INPUT"joueur vers quel lieu"; joueur
5000 GOTO 1140
```

Obtenir des informations sur notre monde devient de ce fait possible, pourtant la nature du mouvement n'est en aucune façon satisfaisante. Nous ne voulons pas sauter sans choisir d'un lieu dans un autre, mais nous souhaitons nous rendre dans une direction voulue.
Normalement le lieu fréquenté par le joueur aura au minimum une, au maximum six sorties. Le lieu 2 se trouve à l'est du lieu 1, aucun chemin ne mène vers le nord, le sud et l'ouest, non plus que vers le haut et vers le bas. Si nous décidons que les directions cardinales seront toujours nommées dans l'ordre N, S, O, E, Haut et Bas, le lieu 1 peut Etre spécifié comme suit :

```
1 dans la forêt     -,-,-,X,-,-
```

Les sorties ne sont alors pas marquées par un X, mais le lieu à atteindre dans cette direction est représenté explicitement.

```
1 dans la forêt     0,0,0,2,0,0
2 dans la forêt     0,0,1,3,0,0
```

Ces six valeurs sont mémorisées pour chaque lieu dans une variable que nous appellerons PASSAGE. Comme deux valeurs différentes (lieu et direction) caractérisent ici le lieu dans lequel on peut pénétrer, la mémorisation dans un tableau bidimensionnel est tout indiquée. La ligne s'apparente chaque fois au lieu, les colonnes un à six correspondent aux chemins possibles.

LIEU	N	S	O	E	H	B
1	0	0	0	2	0	0
2	0	0	1	3	0	0
3	0	4	2	0	0	0
4	3	0	5	0	0	0
5	0	0	6	4	0	0
6	0	0	0	5	0	0

Personne ne devrait éprouver de difficultés à programmer cette table des directions (appelée également Travel-Table). Les débutants qui ne comprendront pas les lignes qui suivent sont priés de revoir l'exemple de la casse donné précédemment.

```
501 PASSAGE(1,1)=0 : PASSAGE(1,2)=0
502 PASSAGE(1,3)=0 : PASSAGE(1,4)=2
```

503 PASSAGE(1,5)=0 : PASSAGE(1,6)=0

Les lignes ci-dessus programment le passage du lieu 1 au lieu 2 en direction de l'est. Nous pouvons agir de façon correspondante avec les autres lieux, mais de cette manière nous gâcherions beaucoup de place en mémoire (combien de fois écrivons-nous 'Passage' ?), de plus l'immense travail de frappe au clavier n'est guère réjouissant. Pour cette raison nous renonçons aux lignes 501 à 503, et nous préférons à la place nous servir des ordres DATA et READ.

DIGRESSION : READ DATA

Habituellement nous utilisons l'instruction INPUT pour transmettre par le clavier des données à un programme en cours. Elle a la forme INPUT Var, Var pouvant être une variable de votre choix. Pour l'entrée séquentielle de plusieurs données, un nombre suffisant de variables séparées par des virgules peuvent être mises à la suite: INPUT LONGUEUR, LARGEUR, HAUTEUR peut ainsi transmettre trois chiffres à la suite au programme.

Si les données sont connues dès le début du programme, elles peuvent être incorporées directement dans ce programme, ce qui n'est valable que si ces données sont toujours les mêmes à chaque exécution du programme. Les mots clef nécessaires à cela sont READ et DATA. INPUT est remplacé par READ: READ LONGUEUR, LARGEUR, HAUTEUR exécute de ce fait la même fonction, les données en entrée sont préparées dans une ligne de programme quelconque avec DATA 1,2,3.

Il est important de savoir que les données sont lues dans l'ordre où elles apparaissent, le nombre des données en lignes DATA doit pour cette raison obligatoirement correspondre au nombre des variables en instructions READ, sinon un message d'erreur apparaît ou bien toutes les données ne sont pas prises en considération.

Si les données doivent être recueillies en commençant à nouveau par le premier élément, il faut utiliser l'ordre RESTORE. Nous entrons donc les lignes suivantes dans notre 464:

```
500 REM**********DESCRIPTIONS DE LIEUX
501 DATA"dans la forêt"
502 DATA"dans la forêt"
503 DATA"dans la forêt devant une paroi rocheuse"
504 DATA"dans une clairière"
505 DATA"dans une clairière"
506 DATA"devant la galerie d'une mine"
```

Maintenant les données doivent être transmises aux variables de travail. Ceci s'effectue avec READ lieu$(lieu), lieu devant naturellement revêtir l'un après l'autre les numéros de lieu un à six. Pour éviter des erreurs, le nombre de lieux doit être connu; nous introduisons la variable AR =

nombre de lieux. Pour bien faire, elle devrait être initialisée dès le début du programme, afin que des extensions ultérieures du jeu puissent être facilement exécutées.

```
110 ar=6
```

Les descriptions de lieux sont entrées à l'intérieur d'une boucle:

```
845 FOR lieu=1 TO AR
850 READ lieu$(lieu)
870 NEXT lieu
```

On introduit de la même façon notre table des directions; pour conserver une vue d'ensemble, nous écrivons les valeurs de direction dans les lignes data correspondantes, à la suite des descriptions de lieu:

```
500 REM ********** DESCRIPTIONS DE LIEU
501 DATA"dans la forêt",0,0,0,2,0,0
502 DATA"dans la forêt",0,0,1,3,0,0
503 DATA"dans la forêt devant une paroi rocheuse",0,4,2,0,0,0
504 DATA"dans une clairière",3,0,5,0,0,0
505 DATA"dans une clairière",0,0,6,4,0,0
506 DATA"devant la galerie d'une mine",0,0,0,5,0,0
```

Les issues possibles doivent être ainsi entrées en même temps que les descriptions. Etant donné que six données de direction suivent chaque lieu, on construit une deuxième boucle à l'intérieur de la première:

```
855 FOR direction=1 TO 6
860 READ passage(lieu,direction)
865 NEXT direction
```

Comme cela, chaque ligne de notre tableau correspond à un lieu; dans les six colonnes de chaque ligne est mémorisé le lieu dans lequel le joueur peut pénétrer en entrant la bonne direction. Ce système nous permettra de programmer facilement la progression du joueur. Pour que le joueur puisse choisir d'aller dans l'une ou l'autre direction, nous devons développer quelques lignes de programme qui impriment en toute lettre sur l'écran les directions possibles. Supposons que notre personnage se trouve en ce moment dans le lieu 3. C'est donc la ligne 3 de notre table des directions qui est concernée:

LIEU	N	S	O	E	H	B
1	0	0	0	2	0	0
2	0	0	1	3	0	0
3	0	4	2	0	0	0
4	3	0	5	0	0	0
5	0	0	6	4	0	0
6	0	0	0	5	0	0

Tout ce que nous devons faire est de sortir les noms des colonnes qui ne

contiennent pas de 0; en l'occurrence, les colonnes deux et trois, sud et ouest. Comme nous l'avons constaté, la ligne du tableau correspond au lieu à examiner; pour conserver les numéros de lieu actuels, nous avions introduit auparavant la variable JOUEUR. Pour sortir les directions non fermées, notre programme doit ainsi tester les six directions du lieu joueur et imprimer les noms de toutes les colonnes qui ont une valeur différente de 0:

```
1250 FOR direction=1 TO 6
1260 IF passage(joueur,direction)<>0 THEN PRINT"**SORTIE**"
1310 NEXT direction
```

Lors d'une exécution test, notre programme décrira maintenant chaque lieu et désignera également chaque sortie possible; mais malheureusement pas encore la direction de celle-ci. C'est pourquoi nous préparons un autre tableau avec les noms des directions (direction$):

```
1020 DATA le Nord, le Sud, l'Ouest, l'Est, en haut, en bas
1030 FOR direction=1 TO 6
1040 READ direction$(direction)
1050 NEXT direction
```

Les noms des six directions sont mémorisés dans les différents éléments et nous pouvons nous représenter le tableau DIRECTION$ comme une sorte de modèle qui est déposé pendant l'exécution du programme sur la ligne concernée de notre table des directions:

```
1240 PRINT"Je peux aller vers ";
1260 IF passage(joueur,direction)<>0 THEN PRINT direction$(direction)
```

Pour le tester, nous démarrons notre programme et entrons quelques numéros de lieu - un coup d'oeil sur notre carte nous assurera que nous avons jusqu'à maintenant tout fait comme il faut. Après ces préparations nous allons rendre maintenant possible le mouvement recherché.

Pour simplifier la vie du joueur, nous n'allons pas lui demander, comme certains jeux d'aventure, d'entre VA OUEST, mais nous n'exigerons que la première lettre de la direction voulue:

```
1390 INPUT"Que dois-je faire";entrée$:entrée$=UPPER$(entrée$)
```

Avant que notre programme exécute maintenant une action du joueur, il devrait en premier lieu vérifier la longueur de l'entrée. Si l'entrée fait plus de deux caractères, il s'agira d'une action quelconque, sinon la routine de mouvement doit être exécutée. Elle teste d'abord si le chemin menant dans la direction souhaitée est libre, si oui, le nouveau lieu est lu dans la table des directions sous la direction correspondante (colonne) du lieu dans lequel le joueur se trouve (ligne), et ce lieu est affecté à la variable correspondante (joueur). Enfin, le joueur est informé de l'exécution de l'action:

```
1080 PRINT
```

```
1400 IF LEN(entrée$)>2 THEN 1500
1410 IF entrée$="N" AND passage(Joueur,1)<>0 THEN Joueur=passage(Joueur,
1):PRINT"O.K.":GOTO 1080
1420 IF entrée$="S" AND passage(Joueur,2)<>0 THEN Joueur=passage(Joueur,
2):PRINT"O.K.":GOTO 1080
1430 IF entrée$="W"AND passage(Joueur,3)<>0 THEN Joueur=passage(Joueur,
3):PRINT"O.K.":GOTO 1080
1440 IF entrée$="O" AND passage(Joueur,4)<>0 THEN Joueur=passage(Joueur,
4):PRINT"O.K.":GOTO 1080
1450 IF entrée$="H" AND passage(Joueur,5)<>0 THEN Joueur=passage(Joueur
,5):PRINT"O.K.":GOTO 1080
1460 IF entrée$="U" AND passage(Joueur,6)<>0 THEN Joueur=passage(Joueur,
6):PRINT"O.K.":GOTO 1080
1470 PRINT"AUCUN CHEMIN N'Y MENE !":GOTO 1080
1500 REM a partir d'ici plus tard analyse de l'entree
```

En commençant chaque jeu, nous devons informer le programme du lieu de départ:

```
150 Joueur=1
```

Une brève promenade dans notre monde du jeu d'aventure nous assurera maintenant rapidement que la chose suivante à entreprendre consiste absolument à organiser l'image du moniteur sinon nous n'aurons sous les yeux, après avoir joué quelques coups, qu'une pagaille d'informations.

FORMATAGE DE LA SORTIE SUR ECRAN

Pour obtenir une représentation nette à l'écran, celui-ci doit bien entendu être effacé au début du jeu:

```
1070 CLS
1080 PRINT:PRINT
```

Ces lignes vides n'ont sur un écran vide bien entendu aucun sens. Elles ne sont toujours indispensables qu'après l'exécution d'un ordre, car l'entrée du joueur comme aussi d'éventuelles informations à celui-ci, doivent glisser d'une ligne vers le haut. Nous obtenons ce scrolling par un PRINT dans les dernières lignes de l'écran et il est indispensable pour terminer chaque coup du jeu. Ce n'est qu'avec la deuxième instruction Print que nous obtenons une ligne vide qui contribue à la clarté de l'image. Se pose alors à nous la question de savoir comment nous parviendrons, après la sortie des informations destinées au joueur, du tiers supérieur de l'écran à la dernière ligne, car finalement nous n'allons pas occuper toutes les lignes intermédiaires.

DIGRESSION: ADRESSAGE DE L'ECRAN

En considérant ses possibilités, le Basic Locomotive du CPC nous offre

deux choix.
D'un côté on trouve les ordres de commande du curseur, grâce auxquels la marque d'écriture peut être déplacée d'un certain nombre de positions à l'écran dans les quatre directions et ainsi également vers le bas. Pour cela le nombre de lignes à sauter doit cependant être connu; mais celui-ci variera pour chaque coup du jeu étant donné qu'il dépend de la quantité d'informations sorties, c'est-à-dire du nombre d'objets.
C'est pourquoi ce chemin ne sera praticable qu'à l'aide de longs calculs!

Comme nous utiliserons plus tard l'un de ces ordres de commande, nous devons aborder rapidement ici les codes de commande.

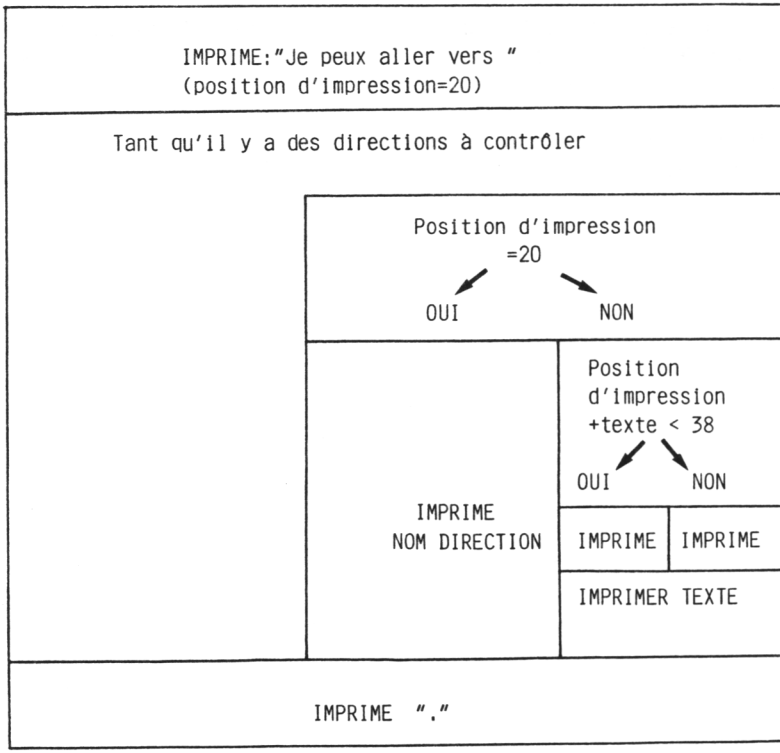
Pour simplifier on peut dire que, pour l'interpréteur Basic, chaque ordre existe sous la forme d'un code numérique, ce qu'on appelle TOKEN, et non pas sous la forme d'une succession de lettres.
Toutes les routines de l'interpréteur prévues pour la gestion de l'écran peuvent être appelées à l'aide de la fonction CHR$() grâce à ce code, les ordres de commande de curseur simples étant répartis comme suit:
CHR$(8) Curseur gauche CHR$(9) Curseur droite
CHR$(10) Curseur bas CHR$(11) Curseur haut
Par exemple, l'instruction PRINT CHR$(11) déplacerait le curseur d'une ligne vers le haut. Pour résoudre notre problème comme du reste presque tous les autres problèmes concernant l'adressage de l'endroit d'écriture, l'ordre 'LOCATE', qui place le curseur à une position bien définie, est nettement plus pratique. On l'utilise sous la forme LOCATE X,Y, où Y correspond à la ligne et X à la position d'écriture à l'intérieur de cette ligne:
```
1390 LOCATE 1,25:INPUT"Que dois-je faire";entrée$:entrée$=UPPER$(entrée$)
```
Nous entrerons donc nos règles du jeu ligne 25 et le RETURN final poussera tout le contenu de l'écran d'une ligne vers le haut, c'est pourquoi pour la communication des réactions au joueur, aucune autre mesure n'est plus nécessaire. Avec le début du coup suivant, la ligne 1080 dégage la ligne inférieure pour le cycle d'entrée suivant.
Malheureusement, dans l'état actuel de développement de notre programme, toutes les sorties seront faites à la ligne inférieure de l'écran, chose qui n'était en aucune façon souhaitable pour la sortie des descriptions de lieu:

```
1130 LOCATE 1,1
```

L'apport d'informations au joueur sur les issues libres ne sera en effet pas si simple, car notre façon de procéder jusqu'alors ne permet pas de

sortir sans problème plus de trois chemins.
Car aucun mot ne devrait dépasser la fin de la ligne, de même les noms des différentes directions devraient être séparées par une virgule et la phrase devrait se terminer par un point.

Développons alors à l'aide de l'exemple suivant une routine qui vient à bout de ce problème d'après le schéma ci-dessous.



Je peux aller vers le nord, le sud, l'est,
en haut, en bas.

Il est sûr qu'avant qu'aucune direction ne puisse être imprimée, 'Je peux aller vers' doit apparaître sur le moniteur (1240).

Pour pouvoir maintenant tester les différentes possibilités a l'intérieur de la boucle, nous devons tout d'abord modifier la ligne 1260:

```
1260 IF passage(joueur,direction)=0 TH
EN GOTO 1310
```

La première issue ne nous posera aucune difficulté, elle peut être imprimée sans tout le travail préparatoire. Nous devons pouvoir distinguer cette première sortie sur écran de toutes les suivantes, ce qui est également possible sans trop de problèmes. Alors la première indication de direction apparaîtra toujours après 'Je peux aller vers' à la vingtième position d'écriture, ainsi:

```
1270 IF POS(#0)=20 THEN PRINT direction$(
direction);:GOTO 1310
```

Avant de sortir toutes les indications de direction suivantes, nous devons toutefois nous assurer tout d'abord que le texte rentre bien encore dans la ligne où l'on doit justement écrire. Si c'est le cas, imprimons d'abord une virgule puis la direction:

```
1280 IF POS(#0)+LEN(direction$(direction))
<38 THEN PRINT", ";direction$(direction);:
GOTO 1310
```

Si la longueur totale de la sortie devait dépasser 38, la sortie d'une virgule tout comme la mise en place de la ligne suivante s'effectuent à l'aide d'une instruction PRINT:

```
1290 PRINT",":PRINT direction$(direction);
:GOTO 1310
```

Grâce à ces trois lignes nous avons couvert toutes les possibilités. Le point final n'est en tout cas nécessaire qu'une seule fois et ne sera mis pour cette raison qu'après abandon de la boucle:

```
1300 NEXT direction
1320 PRINT"."
```

Peut-être est-il encore nécessaire d'expliquer pourquoi nous contrôlons si la longueur de ligne ne dépasse pas 38 caractères, alors qu'une ligne d'écran va de 1 à 40. Eh bien, vous trouvez la réponse aussi bien en ligne 1320 que dans la virgule à la ligne 1280.

En définitive nous devons imaginer une autre possibilité qui n'est pas seulement théorique mais qui trouve parfois, avec les jeux d'aventure difficiles presque toujours, une application pratique.
Je veux parler de lieux qui n'nt apparemment aucune sorte de connexion avec le reste du monde, de sorte qu'aucune direction n'est sortie.
Dans un cas de ce genre, l'image apparaissant sur le moniteur, au stade actuel de notre programme, ne serait pas jolie.
'Je peux aller vers' est certes sorti, mais ensuite vient un point et la solution la plus correcte serait bien 'nulle part'.
C'est pour cela que nous devons nous assurer, avant que notre programme ne se poursuive avec la prochaine action, que quelque chose a déjà été sorti, ce qui peut être fait au moyen d'une autre variable, la variable de contrôle 'dejimp':

```
1240 PRINT"Je peux aller vers ";:dejimp=0

1260 IF passage(joueur,direction)=0 TH
EN GOTO 1310 ELSE dejimp=-1

1310 IF dejimp=0 THEN PRINT"nulle part
";
```

Au début de la partie de sortie, 'dejimp' est fixé à zéro. Si aucune direction n'est par la suite sortie, la ligne 1310 imprime alors le 'nulle part' désiré.

OR, ARGENT ET AUTRES CHOSES UTILES

Une ballade dans le monde de notre jeu d'aventure donne jusqu'ici à l'aventurier une impression vraiment désolante et inintéressante s'il n'a à sa disposition aucun objet pour agir. En outre, même avec la meilleure volonté, de simples descriptions de lieu ne permettront pas du tout de jouer. Saisissons à nouveau notre carte et enrichissons notre monde des objets souhaités, tout en nous limitant en principe à des objets importants pour le déroulement du jeu. Car ajouter des objets pour le plaisir entraînerait plus tard une perte de temps trop importante. En effet, nous devons garantir que le joueur puisse prendre et examiner tout ce qu'il trouve. Cependant nous ne pourrons faire autrement que de placer au moins un objet dans chaque lieu, à moins que l'aventurier ne voie rien, ce dont on peut quand même douter. Si l'imagination nous manque, l'objet en question est justement 'rien de particulier', une formulation que nous trouverons donc très fréquemment dans les jeux d'aventure, parce qu'elle diminue la somme de travail étant donné que dans une forêt normale arbres, buissons, herbe, insectes etc., ne représentent vraiment rien de particulier.
Souvent on ne peut renoncer cependant à des choses inutiles, en particulier quand elles appartiennent à un certain milieu et que le jeu d'aventure doit donner une impression réaliste.

Ainsi, nous aussi nous mettrons des arbres dans notre forêt, bien que l'action ne l'exige pas. Dans le lieu deux, qui se trouve tout près de la montagne, quelques fragments de rochers renforcent l'impression de réalité. De plus, une baraque fait partie du décor d'une mine, peut importe s'il s'agit là de la cabane du mineur ou de sa remise à outils. Pour finir nous devons cacher aussi quelque part les trésors à découvrir,

des objets supplémentaires feront l'affaire qui gêneront plus le joueur qu'ils ne lui seront utiles.
Laissez-moi alors s'il vous plait réaliser de façon plus concrète dans ce qui suit le déroulement d'une action que me suggère mon imagination et je choisirai aussi les objets nécessaires:

DEROULEMENT DE L'ACTION 'L'IVRESSE DE L'OR'

La miniversion donne au joueur pour tâche de trouver deux trésors, des pièces d'or et d'argent aux alentours de la mine. Au début du jeu, le joueur doit se trouver dans la forêt et se rapprocher comme il peut de la mine. En chemin, il découvre dans la forêt un trou dans la terre au fond duquel repose un lourd coffre en fer. Dedans se trouve l'argent, mais malheureusement le coffre est cadenassé et fermé par une lourde chaîne en fer; la clef n'est pas là. Mais à l'entrée de la mine gisent quelques barres de fer assez solides pour faire éclater la chaîne ou le cadenas. Nous pouvons permettre aussi au joueur de tenter cela avec des explosifs qu'il devra trouver dans une cabane en bois. Ce sera pour nous une bonne occasion de programmer sa fin prématurée. Nous déposerons les pièces d'argent dans une caverne habitée, au grand dam du joueur, par un ours féroce. Cela, le joueur ne le saura que quand il examinera la caverne. Si la soif de métal jaune devait vaincre et pousser l'aventurier à mettre tout de suite la main sur l'or, la faim ferait oublier à l'ours sa peur de l'être humain: après la dynamite, le deuxième piège tendu à l'aventurier.

D'autre part, nous savons tous que l'on peut faire très plaisir à un ours avec du miel. Pour cette raison, préparons sur une étagère dans la cabane un pot contenant la précieuse marchandise.
Si le joueur pénètre ainsi armé dans la grotte, l'ours sentira le miel et disparaîtra avec le pot dans les profondeurs de la grotte, ce qui permettra une fin victorieuse pour ce mini-jeu d'aventure.

La carte de notre jeu d'aventure 'L'ivresse de l'or' devrait ressembler maintenant au schéma précédent.

LES OBJETS

Avant de compléter notre jeu avec la programmation des objets, quelques réflexions complémentaires s'avèrent nécessaires.
Pour faciliter au joueur son accoutumance dans notre monde informatique, nous ne décrirons pas ce monde avec des mots secs, mais nous nous efforcerons de maintenir l'intérêt du joueur grâce à des descriptions esthétiques. L'information lapidaire 'Je vois un ours.' permet certes au joueur de jouer, par contre, 'Je vois un ours à l'allure extrêmement féroce' le fera rentrer dans notre jeu.
Si le désir de se faire un nouvel ami s'éveillait chez l'aventurier, une entrée comme 'Caresse l'ours à l'allure extrêmement féroce' serait pour le programmeur que nous sommes certes très simple à réaliser, mais on ne

peut attendre cela du joueur.
De ce fait nous attribuerons à chaque objet, outre sa description propre, un nom d'appel grâce à quoi nous avons la possibilité de fixer librement la longueur des mots de notre jeu selon la nécessité. Ainsi, les jeux d'aventure professionnels des pays de langue anglaise ne demandent la plupart du temps que les trois ou quatre premières lettres d'un mot pour l'identifier. Trois lettres suffisent également en règle générale pour les jeux d'aventure en français, c'est seulement lors de la réalisation d'un plus grand projet que vous devrez au préalable établir une liste des objets prévus et consulter celle-ci attentivement, sinon vous pourriez avoir la mauvaise surprise de constater lors de la programmation des conditions de jeu et des actions que de nombreux mots importants pour un jeu donné commencent par les mêmes lettres.

Comme troisième information en plus du nom complet et du nom abrégé, nous devrons mémoriser pour chaque objet le numéro de lieu où l'objet se trouve au même moment. Etant donné que ces chiffres varient constamment au cours du jeu, ils ne peuvent être lus à partir d'une ligne de programme définitivement programmée, mais ils doivent se présenter sous la forme de variables correspondantes.
Dressons alors une liste dans laquelle chaque enregistrement correspond à l'endroit où se trouve un objet.
Dans ce but nous utiliserons la variable indexée OB() et nous donnons ainsi un outil facile pour pouvoir contrôler et manipuler la position de chaque objet.
Car aussi longtemps qu'un objet ne participe pas activement au jeu, OB=0, nous disons que l'objet se trouve dans l'entrepôt. Dans le cas contraire, le contenu de OB() correspond soit à la valeur de la variable joueur et l'objet en question est ainsi visible dans le lieu, soit alors, égal à -1, il est dans le cas où l'aventurier le prend avec lui (Inventory).

Les lignes suivantes présentent toutes ces informations données sur les objets nécessaires à la réalisation de notre projet de jeu (Ivresse de l'or, miniversion).
Après la description détaillée suit l'abréviation qui est comparé avec les entrées du joueur, puis le numéro de position qui attribue à chaque objet un lieu.
Les guillemets ne sont du reste absolument nécessaires que lorsqu'une virgule est réutilisée à l'intérieur même de la description.

```
300 REM********** OBJETS
301 DATA"combien de grands arbres","arb",1
```

```
302 DATA"combien de grands arbres","arb",2
303 DATA"quelques blocks de rocher","blo",2
304 DATA"une cabane en bois délabrée","cab",3
305 DATA"une bouteille en osier poussiéreuse"
,"bout",0
306 DATA"miel","mie",0
307 DATA"une caisse en bois","cai",3
308 DATA"une étagère branlante","étag",0
309 DATA"quelques explosifs","expl",0
310 DATA"un trou sombre dans la terre","tro",0
311 DATA"un coffre en fer rouillé","cof",0
312 DATA"*des pièces d'argent*","arg",0
313 DATA"une grotte sombre","gro",5
314 DATA"un ours à l'allure féroce","ours",0
315 DATA"beaucoup de petits buissons","bui",4
316 DATA"plusieurs barres de fer","bar",6
317 DATA"*pepites*","pepi",5
```

Comme pour les descriptions de lieu, nous devons fixer ici aussi le nombre d'éléments de données avant d'initialiser les variables:

```
120 ao=17
830 FOR objet=1 to ao
 835 READ ob$(objet),nom$(objet),ob
(objet):nom$(objet)=UPPER$(nom$(ob
jet))
840 NEXT objet
```

```
190 DIM lieu$(ar),passage(ar,6),ob$(ao
),nom$(ao),ob(ao)
```

Pour mémoriser tous les objets, 17 variables différentes sont nécessaires pour lesquelles la place en mémoire nécessaire est réservée par l'instruction DIMension à la ligne 190
Comme pour la majorité des ordinateurs, notre CPC réserve lui aussi en début de programme onze éléments par tableau, c'est pourquoi nous n'avions pas obtenu de message d'erreur malgré l'absence de cette ligne lors de l'initialisation des lieux.

Installons maintenant un lieu de façon visible pour le joueur:
Dans la variable JOUEUR nous trouvons le numéro du lieu dans lequel il entre, OB() contient les numéros de lieu de tous les objets. Vérifions

alors à l'intérieur d'une boucle si le numéro de position d'un objet est bien celui du numéro de lieu actuel:

```
1160 PRINT"Je vois";
1170 FOR i=1 TO ao
1180 IF ob(i)<>Joueur THEN 1210
1190 UF OIS(#0)+LEN(ob$(i))+"<=39 THEN
PRINT ob$(i);", L";:GOTO 1210
1210 NEXT i
```

La ligne de programme 1180 contrôle si chaque objet se trouve dans un autre lieu que le joueur. Si cela s'avère positif, l'objet suivant est aussitôt contrôlé. Sinon la description de l'objet concerné est sortie en ligne 1190. Le point-virgule empêche qu'une nouvelle ligne de sortie soit sas utilisée pour chaque objet supplémentaire. Pour éviter qu'une sortie dépasse la ligne, nous complétons comme suit:

```
1200 IF POS(#0)+LEN(ob$(i))+2>39 THEN PR
INT: GOTO 1190
```

Avant de sortir l'objet suivant, on calcule toute la longueur de l'impression. Si, aprè avoirs imprimé la description de l'objet, la virgule et l'espace entre deux objets (d'où +2), le curseur atteignait une position plus grande que la colonne 39, un retour à la ligne serait effectué. Après quoi le programme saute de nouveau à la ligne 1190, la condition est alors remplie et la description est imprimée.
Toutefois, la virgule derrière le dernier objet est déplacée, c'est pourquoi nous déplacerons le curseur de deux positions vers la gauche et mettrons un point au-dessus de la virgule:

```
1220 PRINT CHR$(8);CHR$(8);"."
```

Pour conclure, n'oublions pas la possibilité qu'il n'y ait aucune donnée à communiquer au joueur et que la phrase 'Je vois' doit être complétée dans ce cas par 'rien de particulier'.
Le principe est connu depuis, de sorte que les compléments nécessaires peuvent être listés ici sans aucun commentaire:

```
1160 PRINT"Je vois";:dejimp=0
1190 IF POS(... ...:GOTO 1205
 1205 dejimp=-1
1215 IF NOT dejimp THEN PRINT"rien de particulier"
```

Nous sommes maintenant vraiment très près de notre but, ce qui nous manque encore pour atteindre la structure d'écran des jeux d'aventure américains originels, c'est une ligne de séparation et une ligne vide:

```
1010 ligne vide$=STRING$(40," ")
1230 PRINT ligne vide$;
1330 PRINT"----------------------------
----------"
```

Nous ne nous apercevrons d'un dernier problème qu'après plusieurs sorties. Les informations à l'intérieur de la moitié inférieure de l'écran avancent toujours vers le haut et se mélangent après quelques coups avec les descriptions de l'environnement. C'est pourquoi le programme doit effacer les lignes supérieures de l'écran avant leur sortie, ce qui peut être effectué en imprimant une quantité suffisante de lignes vides:

```
1090 LOCATE 1,1
1100 FOR ligne=1 TO 10
1110 PRINT lignevide$
1120 NEXT ligne
```

LE VOCABULAIRE

Avant de programmer maintenant le jeu à proprement parler et de nous intéresser aux actions du programme non visibles par un joueur, nous devons entrer rapidement les verbes nécessaires. Avec les noms d'appel des objets évoqués précédemment, ils représenteront le vocabulaire complet de notre jeu d'aventure et seront ainsi à la base du dialogue avec le joueur.
Outre une série de verbes toujours utiles tels que 'Examine', 'Prends', 'Pose', ils dépendront eux aussi du contenu du jeu considéré. L'entrée s'effectue en fonction des lieux et des objets, tout en considérant qu'une abréviation correspondant à la longueur des mots du jeu est suffisante:

```
130 av=6
190 DIM lieu$(ar), passage(ar,6),ob$(ao
),nom$(ao),ob(ao),verbe$(av)

200 REM************************VERBES
```

```
201 DATA exa,pren,pose,ouvr,uti,dét

810 FOR i=1 TO av
815 READ verbe$(i):verbe$(i)=UPPER$(verbe$(
I))
820 NEXT i
```

En outre, nous pouvons donner à l'image avec quelques petits compléments un aspect plus attrayant. Des couleurs différentes nous aiderons à distinguer pendant le jeu les différentes informations. J'ai choisi un fond noir sur lequel les descriptions du monde doivent nous être sorties en bleu foncé. Les chemins libres comme les informations qui s'y rapportent sortent en bleu clair; nos entrées sont faites en blanc.
Bien entendu, à vous d'utiliser des couleurs à votre goût.
A cet effet, vous n'avez à modifier seulement que la deuxième valeur avec l'instruction INK à la ligne 10.

```
10 PAPER 0:INK 0,0:BORDER 0:PEN 1:INK1,
2:PEN 2:INK 2,14:PEN 3:INK 3,26

1130 LOCATE 1,1:PEN 1
1230 PRINT ligne vide$:PEN 2
1290 PEN 3:LOCATE 1,25:INPUT"Que dois-je
faire";entrée$:entrée$=UPPER$(entrée$
):PEN 1
```

Ainsi nous avons effectué la configuration extérieure décrite dans le premier chapitre de cet ouvrage et nous pouvons nous consacrer aux actions de jeu de notre programme.

ANALYSE DES ENTREES

Les changements de direction souhaités par le joueur, un groupe d'instructions avec des mots longs de 1 à 2 caractères ont été déjà vus et correctement exécutés. L'entrée standard se compose donc toujours d'un verbe et d'un objet, les longueurs de mots, sans prendre en compte la longueur de mot minimale nécessaire, ne sont soumises à aucune restriction. Pour pouvoir réagir comme le souhaite le joueur, notre programme doit vérifier ses entrées:

le verbe utilisé est-il programmé ?

l'objet est-il prévu ?

Si aux deux questions il peut être répondu positivement, l'entrée est décomposée en verbe et objet, les numéros de verbe et d'objet sont déterminés; si les éléments de l'entrée ne sont pas définis, une information appropriée est fournie à l'aventurier.

DIGRESSION: TRAITEMENT DES CHAINES DE CARACTERES

La différence principale entre un ordinateur et une calculatrice programmable réside dans l'aptitude de l'ordinateur à traiter des textes. Bien sûr, un ordinateur ne comprend pas ces textes mais elle les conçoit comme une accumulation de signes précis, comme ce que l'on appelle une chaîne de caractères ou string. Il s'agit là en général de combinaisons de tous les signes qui peuvent être entrés entrables par l'intermédiaire du clavier. Le système de fonctionnemt de l'ordinateur met alors à disposition une série de fonctions destinées à traiter ces chaînes de caractères.
Outre une série d'ordres de conversion auxquels appartiennent à part UPPER$ et LOWER$ les fonctions VAL() et STR() qui transforment un chiffre en chaîne ou inversement, on trouve entre autre les instructions LEFT$, RIGHT$ et MID$.
Ces trois fonctions fournissent au programme pour un traitement ultérieur une section de chaîne exactement définie provenant d'une autre chaîne de caractères. Ainsi, l'instruction LEFT$("ABCDEF",3) produit une chaîne composée des trois signes de gauche, à savoir la sous-chaîne ABC.
Il est indispensable de connaître certains paramètres: la chaîne de caractères à traiter et le nombre de lettres souhaité. Ceci est également valable pour RIGHT$.
Par contre, MID$(X$,S,X) nous donne une chaîne tirée de X$ commençant dans l'emplacement S et longue de X signes.
Ainsi notre programme de jeu déterminera par exemple la position de l'espace entre verbe et objet entrés par le joueur et établira à partir de ENTREE$, en utilisant cette marque au moyen des fonctions LEFT$ et RIGHT$, les deux sous-chaînes EVERB$ et EOBJET$.
Pour calculer le nombre de lettres nécessaires, on établit auparavant avec LEN(ENTREE$) la longueur totale de l'instruction de jeu.

Si l'exécution d'un coup du jeu n'est pas faite entre les lignes 1410 et

1470. alors, dans le cas où l'entrée comporte moins de trois caractères, on admet qu'il s'agit d'une erreur d'entrée. Après que 'Aucun chemin n'y conduit' ait été sorti, un nouveau coup commence.
S'il s'agit d'une entrée plus longue, le déroulement du programme se poursuit ligne 2000:

```
1470 IF LEN(entree$)<3 THEN PRINT"Aucun
chemin n'y conduit !":GOTO 1080

2000 longueur=LEN(entree$)
2010 FOR lettre=1 TO longueur
2020 controle$=MID$(entree$,lettre,1)
2030 IF controle$<>" "THEN NEXT lettre
160 longueur de mot=4
2040 everb$=LEFT$(entree$,longueur de mot)
```

Après avoir déterminé la longueur de toute l'entrée, on regarde à l'intérieur d'une boucle si chaque lettre, en commençant par la gauche, est identique à un espace. Une fois l'espace trouvé, la longueur du verbe entré (du premier au dernier signe contrôlé moins un) est déterminée.
Cette valeur, déduite de la longueur totale de l'entrée, correspond au nombre de lettres de l'objet entré.
De cette façon peuvent être déterminés EVERB$ et EOBJET$ en tenant également compte de la longueur de mot propre au jeu.

Si une 'instruction unimot' a été utilisée (HELP), la longueur du verbe sera la même que celle de l'entrée. Etant donné qu'aucun objet ne doit être préparé pour un traitement complémentaire, on saute directement (ligne 2060) à l'analyse du verbe (ligne 2090).

```
2040 everb$=LEFT$(entree$,longueur)
2050 rl=longueur-lettre série de fonctions destinées à traiter ces
chaînes ce caractères.
2060 IF rl<0 THEN 2090
2070 eobjet$=RIGHT$(entree$,rl)
2080 eobjet$=LEFT$(eobjet$,longueur de mot)
2090 FOR numero de verbe=1 TO AV
2100 IF everb$=verb$(numero de verbe) THEN 21
30
2110 NEXT numero de verbe
```

A l'intérieur d'une autre boucle le premier des mots entrés est comparé

aux verbes susceptibles d'apparaître dans le jeu. Si une chaîne identique est trouvée, alors l'indexe de boucle correspond au numero de verbe et la boucle est abandonnée.
Si la boucle a été traitée jusqu'au bout et qu'aucune identité ne fut constatée, c'est que le programmeur n'avait pas prévu le verbe concerné et il en est fait état au joueur :

```
2120 PRINT"Je ne comprends pas le verbe"
":GOTO 1080
```

Pour finir le numéro d'objet est établi selon le même principe :

```
2130 FOR o=1 TO ao
2140 IF eobjet$=nom d'appel$(o) THEN 2200
2150 NEXT o
2160 PRINT"Je ne comprends pas l'objet
 !":GOTO 1080
```

En entrant ces lignes nous avons derrière nous une partie importante de notre travail de développement. Les routines développées jusqu'ici participent à l'élaboration d'une structure intéressante de l'écran de même qu'elles produisent l' 'intelligence' répondant aux entrées du joueur.
Elles déterminent lequel des verbes possibles l'aventurier a utilisé et avec quel objet il souhaiterait faire quelque chose, ce qui permet un saut du cours du programme à une ligne prévue pour l'exécution de l'action voulue. En raison de ces tâches centrales de commande, cette partie de programme s'appelle aussi le moteur.

RECAPITULATION : STRUCTURE DES PROGRAMMES

Ainsi il resort que nos programmes se composent principalement de trois parties :

1. Données du jeu
2. Moteur du jeu d'aventure.
3. Exécution des coups du jeu.

1. Les données représentent la base de tout jeu. Elles sont transmises en partie à des variables de travail ou mises en place à l'intérieur de sous-programmes courts pour la transmission au programme principal.

2. La commande de tout le déroulement du jeu revient au chasseur. Il forme la sortie sur écran, se charge des entrées du joueur, les évalue et lance l'exécution des instructions.
Le moteur est indépendant du jeu d'aventure pratiqué, il peut être utilisé sans modification pour tous les jeux d'aventure.

3. Le jeu d'aventure proprement dit représente la troisième partie du programme. Si toutes les conditions nécessaires à une action sont réunies, alors celle-ci est exécutée.

Afin que vous vous souveniez de cette récapitulation, nous résumons ici la structure de nos jeux d'aventure tout en indiquant déjà les routines restant à déterminer par la suite.
De notre résumé resort aussitôt la possibilité d'utilisation universelle de notre schéma.
Ainsi, quelques petites modifications du moteur suffiront pour faire d'un jeu d'aventure à texte un jeu d'aventure à graphisme. De même, il sera possible grâce à la structure homogène, systématique et aux lignes de programme définies de façon précise pour les divers blocs de fonction, de développer un générateur de jeux d'aventure.

STRUCTURE SCHEMATIQUE: JEU D'AVENTURE

DONNEES

0	100	EN-TETE
100	200	DONNEES DE CONTROLE
		RESERVER PLACE EN MEMOIRE
200	300	VERBES
300	500	DESCRIPTIONS D'OBJETS
500	600	DESCRIPTIONS DES LIEUX
		LIAISONS ENTRE LES LIEUX
600	700	MESSAGES STANDARD
700	800	EVTL. 2. TITRE
800	900	ENTRER DONNEES DU JEU
900	1000	REGLES GENERALES DU JEU

MOTEUR DU
JEU D'AVENTURE

```
1000        1070    INITIALISER LE MOTEUR
1080        1330    GESTION DE L'ECRAN
1331        1389    EVENEMENTS PARTICULIERS
1390                ENTREE DE COUP DU JEU
1391        1399    PIEGES & OBSTACLES
1400        1500    DEPLACER PERSONNAGE PRINCIPAL
1500        1600    INVENTAIRE
1600        1700    SAUVEGARDE DU JEU
1700        1800    CHARGEMENT DU JEU
1800        1950    HELP, VOCABULAIRE, INSTRUCTIONS
1950        2000    INTERRUPTION DU JEU
2000        2160    ANALYSE DES ENTREES
2200                TABLE DE SAUT EXECUTION

EXECUTION

5000        30000   COUPS DU JEU
```

QUELLES SONT LES ACTIONS POSSIBLES?

Imaginons-nous maintenant le joueur qui veut trouver nos deux trésors. D'un seul regard il constate qu'il se trouve au beau milieu d'une forêt, entouré d'arbres et de blocs de rocher. En tant que joueur de jeux d'aventure exercé ou lecteur des chapitres précédents, il sait que l'explication peut résider dans les choses quotidiennes les plus insignifiantes.

Comment réagira-t-il ?
Quelles entrées devons-nous attendre ?

Il essaiera presqu'à coup sûr de continuer avec des instructions telles que EXAMINE FORET, EXAMINE ARBRE, ou EXAMINE BLOCS.
PREND ARBRE est peu probable, cependant nous devons, afin de couper l'herbe sous le pied de tous les critiques de nos programmes, programmer un message du genre 'Je ne suis pas assez fort', en outre, des informations variées montrant qu'une entrée a été très bien comprise peut inciter l'aventurier à persévérer.
C'est normal; si le résultat d'une entrée sur deux ou sur trois est 'Je ne comprends pas ce que tu veux dire !', la tentation d'éteindre l'ordinateur devient alors très forte.

Etant donné que nous n'avons prévu ici encore aucune action mais préférons développer notre monde, nous ne prévoirons pour les lieux 1 et 2 que la sortie de messages divers au joueur, une tâche vite remplie grâce à une instruction print. Pourtant, après avoir pénétré dans le lieu 3 et découvert la cabane en bois, le joueur inspectera celle-ci et découvrira la boonbonne en osier. Cette dernière doit bien sûr, en plus de l'étagère et de la cabane, apparaître à partir de ce moment dans la description de la scène ('Je vois...'), c'est-à-dire, nous devons modifier les lieux où elle se trouve. Un casse-tête supplémentaire pourrait nous être causé par l'instruction PRENDS BONBONNE; dans ce cas la bonbonne à peine apparue devra disparaître aussitôt à nouveau. Elle ne peut revenir à son ancienne place, car si notre joueur fait INVENTAIRE, elle doit naturellement être listée.

Le joueur ne peut évidemment pas examiner la bonbonne en lieu 1 ou en lieu 2, tant il est vrai que toutes les entrées concernant cette bonbonne devraient recevoir une réponse négative aussi longtemps qu'elle ne se trouve pas à proximité du joueur.

C'est de toute façon une affaire de goût qui dépend de l'appréciation du programmeur. Car bien sûr il doit donner l'impression que cette bonbonne se trouve inviolée sur cette étagère depuis des temps immémoriaux et que le joueur ne la voit pas tout de suite.
Si ce joueur est contraint par les nombreuses surprises d'un jeu de recommencer depuis le début, il connaît les positions de départ de différents objets et l'on peut lui permettre sans problème de les prendre même s'ils n'avaient pas été listés dans la ligne 'Je vois...'.

En effet le fait de prendre est même plus simple à programmer, puisqu'il y a une condition de moins à vérifier, cependant nous considèrerons cela comme la majorité des producteurs de jeux d'aventure et ne rendrons également prenables que des objets visibles.

Dans ce cas la cabane en bois de notre personnage principal ne pourra être examinée que si elle se trouve à proximité. Nous ne découvrirons le miel qu'après avoir pris et ouvert la bonbonne. L'ours ne se comportera pacifiquement que lorsqu'il sera occupé à déguster le miel.

Avant d'exécuter une instruction, il faut tout d'abord vérifier que toutes les conditions sont réunies. Même notre bref jeu d'aventure avec seulement six lieux demandera un grand nombre de lignes de programme. Ne considérons que les verbes introduits à ce moment du développement:

examine, prends, pose, ouvre, utilise, détruis, ainsi que nos dix-sept objets, nous devons alors programmer déjà des actions pour 102 entrées différentes. Si nous écrivions les lignes de programme les unes derrière les autres sans aucun schéma particulier, il serait facile de trouver une explication pour les périodes très longues d'exécution du programme. Pour les réduire au minimum, nous diviserons alors la partie de programme à créer en plusieurs blocs.

Nous disposons en principe de deux possibilités différentes. D'une part, nous pourrions réserver pour chaque lieu un nombre suffisant de lignes de programme et y réserver pour chaque possibilité d'action du joueur quelques instructions. Cette technique est très fréquemment choisie, elle

présente aussi l'avantage qu'en raison de l'exécution directe d'une instruction qu'il n'est pas nécessaire d'avoir préparée auparavant, elle est vraiment très rapide, et qu'avant tout un programme sommaire est produit, lequel peut être complété ou édité de façon très aisée, sans vérification dans une quelconque liste. Si vous voulez vous essayer alors à l'exécution d'un jeu d'aventure, la réalisation suivante de notre lieu 1 sera une incitation suffisante. N'oubliez pas qu'il s'agit d'un exemple qui n'a rien à voir avec notre schéma. Donc, prière de ne pas remplacer de ligne du programme que nous avons déjà établi jusqu'à maintenant par ces lignes.

```
100 INPUT"Que dois-je faire";entree$
200 REM entree$ en verbe$ et objet$
analyser (comme dit)
300 ON lieu GOTO 1000, 2000, 3000, 4000,
 5000, 6000
301 REM sauter dans le lieu concerne
en relation avec le numero de lieu

1000 CLS:REM **************lieu 1
1010 PRINT"Je suis dans la forêt."
1020 PRINT"Je vois beaucoup de grands arbres."
1030:
1040 REM imprimer d'autres objets
1100 PRINT"Je peux aller vers le sud, l'est."
1200 IF entree$="S" THEN lieu=6: GOTO 1
00
1200IF entree$="E" THEN lieu=2: GOTO 1
00
1260 IF LEN(entree$)<3 THEN PRINT "Aucun
```

```
chemin n'y conduit !": GOTO 100
1300 IF entree$="INV" OR "INVENTAIRE" THE
N GOTO 200
1400 IF everb$="EXA" AND eobjet$="ARB
" THEN PRINT"Je ne vois rien de particulier.
":GOTO 100
1410 IF everb$="PREND" AND eobjet$="ARB
" THEN PRINT"Je ne suis pas assez fort.":GO
TO 100
1999 PRINT"Je ne comprends pas ce que tu veux dire.": GOTO 100
2000 a partir d'ici traitement du lieu 2
3000 a partir d'ici traitement du lieu 3
```

Si vous avez regardé ou même entré les lignes ci-dessus dans votre CPC, vous constaterez qu'un jeu d'aventure tout à fait viable peut être produit d'une façon encore plus facile, et vous vous demanderez peut-être pourquoi nous n'avons pas utilisé ce schéma où il n'est pas besoin de longs commentaires pour comprendre ?
Eh bien, d'une part des jeux d'aventure particulièrement volumineux demanderont infiniment plus de place de mémoire, car nous ne pourrions éviter de produire plusieurs fois des routines identiques vu que, pour donner un exemple connu, nous plaçons la bonbonne dans chaque lieu (six lignes de programme identiques) et que, ne l'oublions pas, nous devons pouvoir la reprendre. Face à ces douze lignes on trouve dans notre système de jeu d'aventure deux lignes de programme. Cela seul justifie notre choix, mais même l'argument de la clarté d'ensemble et la possibilité d'édition doit être rapidement infirmé.
Imaginez-vous qu'en complétant le jeu d'aventure d'autres bonbonnes soient nécessaires pour la poursuite de l'action, de telle sorte que nous préférions faire, pour une raison quelconque, un petit tonneau de cette bonbonne. Si votre jeu avait déjà à ce moment quarante lieux, vous pouvez faire de 'Prends bonbonne', un 'Prends tonneau' avec un nombre correspondant de lignes. La même chose vaut pour 'Pose bonbonne, ouvre bonbonne' etc. .
Si par contre vous vous en êtes tenu à notre système de jeu, modifiez la ligne 305, et c'est tout!

N'oublions donc pas cette méthode et retournons à notre jeu d'aventure. Nous laissons l'action se dérouler également dans des lignes de programme définies de façon précise, mais chaque bloc contient maintenant le traitement d'un verbe correspondant.

Notre moteur a déjà déterminé le numéro de verbe et d'objet. nous pouvons alors appeler individuellement ces blocs. Il est donc logique de regrouper toutes les lignes qui se chargent de l'exécution de l'instruction 'examine', par exemple les lignes de programme à partir de 5000. l'action 'prends' est alors en conséquence réalisée à partir de la ligne 6000.

Le moteur se charge, à l'aide du numéro de verbe, de la commande du déroulement du programme à la ligne 2000:

```
190 REM                EXA PREN POSE
OUVRE UTILISE DETRUIS
2200 ON numeroverbe GOTO 5000,6000,7000,8
000,9000,10000
```

C'est à la ligne 5000 que commence le bloc 'EXAMINE'. On se rend compte facilement que la plus grande somme de travail nous est réclamée ici.
Non seulement il faut qu'à EXAMINE CABANE l'information DANS LA CABAME SE TROUVE UNE VIEILLE ETAGERE sorte, et que cette étagère devienne un objet visible, mais nous devons aussi nous assurer que la cabane se trouve bien dans le lieu actuellement visité et transmettre le cas échéant une information comme JE NE VOIS RIEN DE LA SORTE ICI au joueur.

Ces réflexions montrent déjà le volume de travail de programmation d'un jeu d'aventure, par bonheur de nombreuses entrées différentes effectuées par le joueur reçoivent une réponse d'une même et unique ligne de programme.
Ainsi notre partie de programme qui traite les actions où le joueur examine vérifie en premier lieu si l'objet nécessitant de plus amples informations se trouve à la portée du joueur, à savoir dans le même lieu ou s'il l'a en sa possession. Dans le cas contraire, il peut être procédé aussitôt à l'entrée de lu coup de jeu suivant:

```
5000 IF ob(o)<>joueur AND ob(o)<>-1 THE
N GOTO 5900
5900 REM       OBJET ABSENT
5990 PRINT"Je ne vois rien de la sorte ici
!":GOTO 1080
```

LES CONDITIONS

Ces deux conditions mises à part, on peut en formuler beaucoup d'autres qui peuvent être considérées comme des critères pour l'exécution d'une action et sans lesquelles un jeu d'aventure ne pourra être réalisé:

L'objet est dans le lieu
L'objet est porté par le joueur
L'objet n'est pas dans le lieu
Le flag est mis
Le flag n'est pas mis
Le joueur est dans un lieu précis

Comme nous le verrons plus tard, cette liste ne prétend pas être exhaustive, on peut imaginer par exemple une inversion de la dernière condition, certaines actions ne peuvent être exécutées dans un certain lieu mais peuvent l'être dans tous les autres lieux.

Les flags nécessitent une explication. Il s'agit là de commutateurs de signaux qui participent directement au contrôle du déroulement du jeu e
t
que nous réaliserons à l'aide d'un autre tableau. Ces variables de taableau ne peuvent aussi admettre que deux positions: soit un flag est mis et le contenu de la variable doit être -1, soit il n'est pas mis ce qui doit correspondre à 0.
En pratique nous l'utiliserons pour identifier des situations:

une porte est-elle ouverte ou fermée à clef ?
un obstacle fut-il évité ou non ?
un monstre doit-il mettre un terme au périple du joueur ou celui-ci a-t-il le droit de continuer ?

Dans la pratique de la programmation, la formulation d'une seule condition ne suffira pas en général pour établir une classification très nette de la situation ce qui rend des recoupements logiques de plusieurs conditions nécessaires pour garantir que certaines actions ne seront pas exécutées de façon inopportune.
Il s'agira pour ces opérations logiques de liaisons ET ou de liaisons OU, cela signifie que dans la pratique les deux liaisons (AND) ou seulement l'une ou l'autre (OR) de deux ou de plusieurs liaisons doivent être remplies. Si nous ne pouvons renoncer à plus de deux conditions, une mise entre parenthèses est en général indispensable. Démontrons cela clairement à l'aide d'un exemple:

Pensons à notre explosif dans notre caisse. Il doit servir à gâter la joie des joueurs par trop zélés. Pour des raisons de fair-play, un avertissement survenant avant l'explosion est toutefois absolument indiqué, c'est pourquoi le message 'il risque d'exploser' doit sortir après 'Examine l'explosif'. Si le joueur tente encore par la suite de s'en saisir, il est lui-même responsable de sa propre fin.

Si le joueur a effectivement fait l'entrée ci-dessus, le moteur mettra les numéros 1 (examine) et 9 (explosif) à disposition. Notre problème réside maintenant dans le fait que nous ne pouvons nous satisfaire de la vérification d'un lieu, car nous ne devons pas oublier que le joueur ne peut certes pas saisir et emmener l'explosif mais qu'il peut le faire avec la caisse en bois. Et s'il emporte cette caisse avec lui, il emporte l'explosif avec. Vérifions donc de ce fait si le joueur comptait emmener l'explosif, ET si en même temps la caisse en bois (objet numéro 7) se trouve dans le lieu visité OU dans l'inventaire du jeu.
L'objet de l'action proprement dit (numéro 9) n'est pas disponible, le cours du programme se poursuivra donc après la ligne 5000 à la ligne 5900.

L'exécution de l'instruction 'Examine explosif' doit alors être formulée comme suit:

```
5904 IF o=9 AND (ob(7)=Joueur OR ob(7)=
-1) THEN PRINT"Il risque d'exploser !":GOTO 1080
```

LES ACTIONS

L'exemple précédent nous a éclairé sur l'exécution de l'action véritablement la plus simple, à savoir la sortie d'une information au joueur.
En général cette instruction-print n'est cependant qu'un accessoire pour d'autres actions comme les manipulations de OB(), pour confirmer au joueur, dans un message en retour, l'exécution de l'instruction.

A l'inverse, des instructions comme Prends, Pose, Détruis, et aussi ouvre signifient toujours pour un objet un changement de lieu. Soit il est de nouveau emmené par le joueur, soit il appartient désormais à l'inventaire d'un certain lieu, il est aussi possible que cet objet disparaisse totalement du jeu. Imaginez un peu la consternation du joueur de l'ivresse de L'Or une fois que l'ours a dégusté le miel, si la bonbonne

était toujours présente partout!

Une autre action également nécessaire de façon fréquente est la mise ou l'annulation des flags déjà cités, pour créer progressivement les conditions pour des actions ultérieures.

Nous ne devons oublier en aucun cas des changements de lieu du joueur provoqués et déclenchés par la magie ou des actions anodines. Une autre extension de 'L'ivresse de l'or' pourrait faire de l'antre de l'ours un gigantesque dédale de galeries, et après 'Examine grotte' le joueur se trouverait au beau milieu de la grotte sans l'avoir voulu et quelle que soit la direction demandée. Mais notre jeu gagnera en réalisme ce qu'il perdra en temps de travail. En effet, comment voulez-vous examiner minutieusement un grotte sans y pénétrer ?

Comme pour les conditions, nous pouvons résumer ici en une liste les actions possibles, liste ne se prétendant pas, répétons-le, exhaustive. Il s'agit là avant tout d'une base de travail minimale, car il est possible d'écrire de bons jeux d'aventure avec les seules conditions et actions présentées jusqu'ici.

Sortie d'information au joueur
L'objet disparaît
L'objet va dans l'inventaire
Un nouvel objet apparaît dans le lieu
Le flag est mis
Le flag est annulé
Le joueur est transporté dans un autre lieu

PROGRAMMATION DE L'EXECUTION DES INSTRUCTIONS

Nous allons maintenant vous donner quelques conseils concernant certaines particularités dont il faut tenir compte pour certaines instructions. Nous n'expliquerons que quelques lignes de programme. Les autres vous sont fournies dans le listing de la miniversion à la fin de ce chapitre.

Action: examine

Le résultat de l'examen d'un objet est toujours fourni par une phrase.

Lorsqu'un message est utilisé plusieurs fois ("Je ne vois rien de spécial"), nous pouvons affecter son contenu à une variable.

```
600 REM ********** MESSAGES
601 m$(1)="Je ne vois rien de special."
602 m$(2)="Je ne suis pas si fort!"
603 m$(3)="Comment vois-tu cela?"
604 m$(4)="L'ours prend le miel "
605 m$(5)="et disparaît dans la grotte."
690 ok$="D'accord!"
```

Nous avons déjà expliqué la ligne 5000 qui vérifie d'après la valeur de OB() que l'objet se trouve à proximité du joueur.
Si c'est le cas, on détermine, en fonction du numéro d'objet o quelle est la ligne qui correspond à cet objet. Si toutes les conditions programmées sont réunies, le reste de la ligne est traité et le déroulement du programme se poursuit ensuite par la reconstitution de l'image de l'écran:

```
5002 IF o=1 THEN PRINT m$(1):GOTO 1080
5003 IF o=3 THEN PRINT m$(1):GOTO 1080
5004 IF o=4 THEN PRINT"Il y a une étagère dans un coin."
:OB(8)=Joueur:GOTO 1080
```

Si la cabane est examinée (o=4), le joueur découvre l'étagère. Celle-ci se trouvait jusqu'à présent dans le lieu 0, l'entrepôt. Elle est maintenant placée dans le lieu du joueur.

Il est possible qu'un objet apparaisse que le joueur emmènera ensuite avec lui.
Il ne faut pas dans ce cas que le joueur retrouve cet objet dans le lieu où il l'a pris.

Nous pouvons éviter cette erreur en nous assurant d'abord que l'objet, par exemple ici la bonbonne posée sur l'étagère, ne se trouvait pas encore dans le jeu et qu'il était encore dans l'entrepôt.

```
5008 IF o=8 AND ob(5)=0 THEN PRINT"Il y a une bonbonne sur l'etagere."
:ob(5)=Joueur:GOTO 1080
5009 IF o=8 AND ob(5)<>0 THEN PRINT m$(1):GOTO 1080
```

Lors d'examens ultérieurs de l'étagère, le joueur ne trouvera rien de

spécial (ligne 5009).

Nous allons maintenant voir un exemple d'utilisation des flags. Si le joueur a trouvé le coffre du trésor, celui-ci est d'abord fermé par une chaîne de fer. Après que cet obstacle ait été contourné, il faut encore ouvrir le coffre avant que son précieux contenu ne devienne visible. Pour pouvoir indiquer au joueur les trois états possibles du coffre, construisez une table définissant la fonction des différents flags:

FLAG	0	-1
1	Chaîne entière	Chaîne détruite
2	Coffre fermé	Coffre ouvert

Nous utilisons les valeurs -1 et 0 pour économiser un peu de travail de programmation. En effet, une expression vraie est représentée par l'interpréteur Basic par -1 et une expression fausse par 0. Nous pouvons donc abréger quelque peu nos instructions IF:

```
IF FL(1)=-1     correspond à   IF FL(1)
IF FL(1)=0      correspond à   IF NOT FL(1)
```

Nous obtenons ainsi:

```
5011 IF o=11 AND NOT fl(1) THEN PRINT"Il
est ferme par une chaine de fer.":GOTO 1080
5012 IF o=11 AND fl(1) AND NOT fl(2) THEN PRINT"Je
ne vois rien de spécial extérieurement.":GOTO 1080
5013 IF o=11 AND fl(1) AND fl(2) AND ob(12)=0 THEN PRINT"Il
est plein de pieces d'argent.":ob(12)=Joueur:GOTO 1080
```

Il nous faut également traiter le problème que constitue la présence de deux objets identiques (1 et 2) dans les deux premiers lieux. Nous pouvons utiliser pour régler ce problème la ligne 5000 dont la fonction est également de traiter les mauvaises entrées.
Voici donc la solution que nous adopterons pour "examine arbres" dans le lieu 2:

```
5901 IF o=1 AND Joueur=2 THEN PRINT m$(
1):GOTO 1080
```

Notre routine doit se terminer par une ligne qui soit exécutée sans

aucune condition, lorsque les conditions précédentes n'ont pas été remplies. Nous éviterons ainsi que soient exécutées les lignes correspondant aux autres verbes, ce qui risquerait de nous faire perdre le contrôle du programme. En outre, le joueur ne remarquera pas si nous avons oublié une action qui n'est pas nécessaire pour le déroulement de l'aventure:

```
5990 PRINT"Je ne vois rien de tel ici!":GOTO 1080
```

Action: PRENDS

De même que le joueur ne peut examiner que lorsqu'il est à proximité, de même il ne pourra le prendre que lorsqu'il est dans le même lieu que le joueur ou qu'il est dans sa poche.

```
6000 IF ob(o)<>Joueur AND ob(n)<>-1 THEN
GOTO 6900
```

Mais il vaut mieux que nous empêchions le joueur d'emporter certains objets. Nous supposerons par exemple qu'un coffre plein est trop lourd pour un seul homme et nous lui interdirons des tentatives stupides du genre "prends arbre":

```
6001 IF o=1 THEN PRINT m$(2):GOTO 1080
6005 IF o=11 THEN PRINT m$(2):GOTO 1080
```

Bien entendu les choses peuvent aussi tourner plus mal, de façon à ce que la victoire ne soit pas trop facile. Par exemple, selon notre scénario, si le joueur veut prendre les explosifs, le simple contact déclenchera une explosion mortelle qui amènera la fin du jeu. De même, s'il s'empare des pépites avant d'avoir offert le miel à l'ours ou s'il essaie de prendre l'ours, cela se terminera mal pour le joueur:

```
6015 IF o=1 AND Joueur=5 THEN m$(0)="L'ours
m'a tue.":GOTO 4500
6018 IF o=17 AND NOT fl(3) THEN m$(0)="Un
ours se precipite sur moi.":GOTO 4500
6900 IF o=9 AND (ob(7)=Joueur OR ob(7)=-1)
THEN m$(0)="Le contact a fait sauter les
explosifs!":GOTO 4500
```

Remarque: en ligne 4500 se trouve la routine qui est appelée lorsque le

joueur a perdu. m$(0) est utilisé par cette routine pour expliquer la raison de l'échec.

Le flag 3 est mis lorsque l'ours a pris le miel.

Si rien ne s'oppose à ce que le joueur s'enrichisse d'un quelconque objet de notre jeu, il nous faut attribuer la nouveau numéro d'emplacement -1 à l'objet correspondant:

```
6010 IF o=5 THEN ob(5)=-1:PRINT ok$:GOTO 1080
```

Action: POSE

Le joueur peut cependant à tout moment vouloir se débarrasser d'un objet qu'il a pris, soit parce que nous avons, par programmation, limité le nombre d'objet que peut porter le joueur, soit parce qu'il veut utiliser un objet.
La seule condition pour que le joueur puisse poser un objet est bien sûr qu'il le porte sur lui:

```
7000 IF ob(o)<>-1 THEN PRINT"Je ne possede
encore rien de tel!":GOTO 1080
7900 ob(o)=joueur:PRINT ok$:GOTO 1080
```

Il se peut cependant qu'il y ait d'autres actions à exécuter lorsque le joueur pose un objet. Nous avons par exemple prévu que l'ours s'empare de la bonbonne de miel pour disparaître avec elle dans la grotte.
Bien entendu cette action ne doit être exécutée que si le joueur se trouve dans le lieu 5:

```
7020 IF o=5 AND joueur=5 THEN ob(5)=0:fl(3)=-1:
PRINT m$(4):PRINT m$(5):ob(14)=0:GOTO 1080
```

Action: OUVRE

Cette section de programme commence par le test qui nous est maintenant bien connu et qui est destiné à éviter des erreurs techniques. Il est à nouveau nécessaire de contrôler le lieu ainsi que l'inventaire. En effet, si on ne demandera pas au joueur de prendre une porte avant de l'ouvrir, cela pourra cependant être une condition à remplir avant d'ouvrir une bouteille:

```
8000 IF ob(o)<>joueur AND ob(o)<>-1 THEN
PRINT"Je ne vois rien de tel ici.":GOTO
1080
```

L'action elle-même consistera le plus souvent à mettre un flag et à sortir un message:

```
8025 IF o=1 AND fl(1) THEN PRINT ok$;
"- le couvercle s'ouvre et retombe en arriere.":fl(2)=
-1:GOTO 1080
```

Nous ne devons pas oublier les actions qui n'ont pas d'importance mais qui sont possibles. Reprenons l'exemple de la bonbonne qu'un aventurier risque de vouloir ouvrir avant qu'il ne l'ait examiné. Il serait surpris que nous l'empêchions d'ouvrir la bonbonne simplement parce que nous trouvons cela inutile dans la logique du jeu:

```
8010 IF o=5 THEN PRINT ok$:GOTO 1080
```

Et nous terminons par une ligne qui nous permettra de déceler les impossibilités physiques comme par exemple "ouvre miel":

```
8999 PRINT"Je ne comprends pas ce que tu veux dire.":GOTO 1080
```

RECAPITULATION

J'espère que grâce aux explications sur les actions les plus fréquentes vous comprenez maintenant l'exécution de la programmation proprement dite. En principe, on peut dire pour résumer que les différentes actions sont entourées par deux lignes de programme.
Ainsi on vérifie au début d'un bloc si l'exécution de l'action est techniquement possible, pour finir on s'assure que le déroulement du programme continue de s'effectuer correctement même lors de l'apparition d'une faute quelconque.
Trouver les justes conditions à l'exécution d'une instruction est cependant, comme l'action elle-même, en général une chose très simple à programmer qui a été de surcroît standardisée dans notre système de jeu d'aventure. Le tableau récapitulatif suivant doit vous aider dans la réalisation des premiers jeux d'aventure de votre création :

CONDITION	PROGRAMME BASIC
L'objet est dans le lieu	ob(objet)=sp
L'objet est emporté par le joueur	ob(objet)=-1
L'objet n'est pas dans le lieu	ob(objet)<>sp
Le flag est mis	fl(x)=-1
Le flag n'est pas mis	fl(x)=0
Le joueur est dans un lieu donné	sp=No de lieu

ACTIONS	PROGRAMME BASIC
Sortir message au joueur	PRINT" ou M$
L'objet disparaît	ob(Objet)=0
L'objet va dans l'inventaire	ob(Objet)=-1
Un nouvel objet apparaît dans le lieu	ob(Objet)=sp
Le flag est mis	fl(x)=-1
Le flag est annulé	fl(x)=0
Le joueur est transposé dans un autre lieu	sp=No de lieu

LES DERNIERS PAS

Avant que notre jeu d'aventure corresponde aux idées développées dans le chapitre 2, nous devons encore réfléchir sur trois autres parties de programme.
Il est ainsi absolument nécessaire pour le déroulement parfait d'un jeu

de donner au joueur la possibilité de faire un rapide inventaire.

Il serait parfaitement possible d'exécuter cette action de la même façon que le traitement des autres verbes. Comme il s'agit ici toutefois d'une fonction fondamentale des programmes de jeux d'aventure, nous compléterons notre moteur d'une façon appropriée.

Inventaire

L'inventaire appartient, comme aussi les routines à construire plus tard pour sauvegarder et charger la situation du jeu, au groupe des instructions unimot. Pour cette partie de programme nous avons laissé entre les lignes de programme 1480 (fin des routines de mouvement) et 2000 (analyse des entrées) suffisamment de place.

Afin de ne pas occuper inutilement notre moteur à comparer longuement chaque coup du jeu avec les instructions de ces groupes particuliers, nous déterminons tout d'abord s'il s'agit d'une de ces instructions spéciales ou d'une entrée régulière.
L'entrée courante du joueur aura une longueur minimale de neuf lettres. Partons du fait, si l'entrée est moins longue, qu'il s'agit d'une instruction unimot :

```
1480 IF LEN(entree$)>8 THEN GOTO 2000
```

S'il ne s'agissait pas dans cette entrée d'une combinaison régulière verbe/objet, toutes les lignes dans lesquelles le traitement d'une instruction unimot commence doivent être parcourues jusqu'à ce que l'on finisse par déterminer une concordance des trois premières lettres.

Dans la pratique, l'exécution de l'inventaire se compose d'une simple boucle qui sort à l'écran tous les objets du jeu dont le lieu est désigné par -1 :

```
1499 REM ********** DEBUT D'INVENTAIRE
1500 IF LEFT$(entree$,3)"INV" THEN GO
TO 2000
1510 PRINT"Je porte la chose suivante avec moi:"
1520 FOR objet=1 TO ao
1530 IF ob(objet)=-1 TNEN PRINT ob$(objet)
1540 NEXT objet
```

```
1550 GOTO 1080
1560 REM ********** FIN D'INVENTAIRE
```

Les titres

D'un point de vue purement technique, avec ces lignes nous terminons notre programme. Toutes les fonctions souhaitées sont disponibles et rien n'empêche plus de jouer. Pourtant, combien de temps l'aventurier devra-t-il errer dans notre monde, à quel moment a-t-il atteint son but ?

Conformément à l'action, la miniversion de L'ivresse de l'or est finie lorsque les pépites et les pièces d'argent se trouvent dans les mains du joueur.
Une seule ligne Basic suffit pour interdire dans ce cas toutes les entrées supplémentaires et appeler une partie de programme qui annonce au joueur sous une forme appropriée sa réussite et met un point final au jeu.

```
1340 IF ob(12)=-1 AND ob(17)=-1 THEN GOT
O 4800
```

Une fin totalement différente, moins glorieuse est cependant fréquente dans les jeux d'aventure. Etant donné que certains joueurs termineront par la force des choses leur jeu de façon quelque peu prématurée, nous programmons une autre image finale à partir de la ligne 4500.
Lors de l'exécution des actions amenant à la fin du jeu, nous avions déjà préparé un message adéquat afin qu'au moins le joueur apprenne les raisons de son échec.

```
4500 CLS:REM                          JOUEUR MORT
4600 PRINT"Meme cela !":PRINT:PRINT
m$(0)
4610 PRINT:PRINT"Je suis mort !":PRINT
4620 INPUT"Dois-je essayer encore une fois "
;entree$:entree$=UPPER$(entree$
)4630 IF LEFT$(entree$,1)="J" THEN RUN
4640 GOTO 1960
```

Ceci nous permet l'intégration du message individuel dans m$(0), et d'utiliser cet entête pour chaque situation et pour chaque jeu d'aventure. Dans la mesure où elles sont absolument nécessaires, des

modifications demanderont peu de travail. Pour cette raison, les lignes ci-dessus ainsi du reste que le message de victoire à partir de 4800 peuvent être considérees comme des compléments de notre moteur.

Pensons en outre qu'il n'est absolument pas courant de charger des jeux d'aventure et d'aboutir en très peu de temps. Nous programmons donc encore et en dernier lieu une possibilité de mettre fin au programme par une instruction spéciale.

Même si l'on trouve de tels logiciels sur le marché, il n'est pas tout à fait normal qu'on ne puisse mettre fin à un programme qu'en utilisant la touche ESC ou les touches RESET.

Introduisons de ce fait une autre instruction unimot dans notre système :

```
1500 IF LEFT$(entree$,3)<>"INV" THEN GO
TO 1950
1950 IF LEFT$(entree$,3)<>"END" THEN GO
TO 2000 ELSE CLS
1960 PRINT "L'auteur vous souhaite plus
de réussite":PRINT"la prochaine fois !"
:PRINT:PRINT:PRINT:END
```

Ceci représentait donc les dernières lignes de notre jeu d'aventure, toutes les versions imaginables de fin du jeu ont été réalisées. Mais qu'en est-il par contre du début de notre oeuvre ?

Si nous commençons notre programme tel qu'il est, sa première action consiste à préparer les variables, un travail qui peut faire croire, surtout pour des jeux plus importants, que le programme est planté. Puis on se retrouve au beau milieu du programme et un novice en la matiére, donc inexpérimenté, comprendra le jeu comme l'organisation d'une visite guidée dans un monde simulé au moyen de l'électronique.

Faisons donc crédit à une vieille croyance populaire qui dit que la première impression est souvent la bonne.
Donnons-nous la peine d'affecter un, voire mieux plusieurs titres à notre programme avant de le considérer comme terminé.
Le premier titre aura pour fonction de donner sous une forme brève mais correcte des renseignements sur les octets se trouvant en mémoire ainsi que d'annoncer le titre et le genre du programme. Il donnera aussi des informations sur l'auteur.

Nous fairons apparaître l'image suivante sur l'écran avant que les variables soient initialisées. Pendant ce laps de temps, ce titre peut donner des renseignements sur la tâche incombant au joueur et le faire pénétrer dans l'action.

```
                Bienvenue à
            "L'IVRESSE DE L'OR"
--------------------------------------------------
A la recherche de la fortune  dans le Nouveau Monde,
vous avez rencontré il y a quelques jours un vieil
homme agonisant que vous avez assisté pendant les
heures qui lui restaient à vivre.
Par reconnaissance, il vous parla de sa mine d'or
et du reste de sa fortune cachée là-bas.
Sur le chemin conduisant à la mine, vous
avez surmonté d'innombrables dangers; vous
atteindrez bientôt votre destination et
vous verrez si le vieux a dit la vérité
ou si la fièvre l'a fait délirer.
--------------------------------------------------
Avez-vous besoin de conseils ?
```

Si la découverte de la tâche à accomplir fait cependant partie de l'idée du jeu, ce tableau n'apparaît pas et au titre du programme s'ajoute aussitôt une explication des jeux d'aventure qui doit être introduite sinon comme troisième en-tête.

Car comment était-ce lorsque vous avez eu votre premier jeu d'aventure dans les mains et en mémoire, connaissiez-vous le sens et le but de ces programmes, ou bien étiez-vous assis devant votre ordinateur perplexe et inactif ?

N'oublions pas qu'il y aura toujours des débutants. Des débutants qui sont contents lorsqu'ils peuvent charger et commencer un programme sans messages d'erreur. Des règles du jeu sommaires ne représentenant pas un énorme travail. Mettons-nous donc à l'ouvrage.

CPC - Ventures
(c) 1984 par Jörg Walkowiak

Imaginez-vous un robot que vous pouvez diriger avec de nombreux ordres.
Je suis ce robot et je m'exposerai pour vous aux dangers des aventures la plus risquées.
Afin que vous puissiez me faire agir intelligemment je vous décrirai à chaque fois de façon très précise la situation dans laquelle je me trouve.
Puis dites-moi avec deux mots comme par exemple EXAMINE PORTE, PRENDS COUTEAU, ce que je dois faire.

En outre je comprends les instructions
INVENTAIRE, SCORE, VOCABLES, AIDE
SAUVEGARDER & CHARGER ainsi que END.

Deux exemples vous ont montré comment cela était conçu. Pour ces tâches sont prévues les lignes de programme 10 - 100 ainsi que 700 - 900.
Les explications générales seront insérées comme sous-programme, ce qui permet la poursuite de la construction du moteur avec l'instruction INStruction. L'aventurier peut maintenant se remettre à tout moment les règles générales en mémoire sans interrompre le jeu.

INDICATIONS POUR LE LISTING SUIVANT

Après avoir entré les lignes suivantes dans votre CPC, vous pouvez disposer de la miniversion de 'L'ivresse de l'or'. Si vous avez entré au fur et à mesure de nos explications les différentes lignes d'exemple, contrôlez qu'elles soient bien conformes au listing suivant.

Vous verrez que grâce aux techniques présentées jusqu'ici, des jeux d'aventure peuvent déjà être développés qui ne souffrent pas de la comparaison avec des logiciels commerciaux.

Il tient alors à vous de poser d'abord ce livre et de développer vous-même un jeu d'aventure ou plutôt de faire connaissance avec d'autres techniques qui peuvent faire d'un jeu d'aventure correct un jeu d'aventure parfait.

```
10 MODE 1:PAPER 0:INK 0,0:BORDER 0:PEN 1:INK 1,25:PEN 2:INK 2,24
110 AR=6
120 ao=19
130 av=6
140 af=3
150 joueur=1
160 longueurmot=4
190 DIM lieu$(ar),passage(ar,6),ob$(ao),nom$(ao),ob(ao),verb$(av)
200 REM ************************ VERBES
201 DATA exam,pren,pose,ouvr,util,detr
300 REM ******************* OBJETS
301 DATA"beaucoup de grands arbres","ARBR",1
302 DATA"beaucoup de grands arbres","ARBR",2
303 DATA"quelques rochers","ROCH",2
304 DATA"une cabane en bois en ruine","CABA",3
305 DATA"une bonbonne sale","BONB",0
306 DATA"du miel","MIEL",0
307 DATA"une caisse en bois","CAIS",3
308 DATA"une etagere branlante","ETAG",0
309 DATA"un peu d'explosif","EXPL",0
310 DATA"un sombre puits naturel","PUIT",0
311 DATA"un coffre de fer rouille","COFF",0
312 DATA"*des pieces d'argent*","PIEC",0
313 DATA"une sombre grotte dans la roche","GROT",5
314 DATA"un ours a l'air mechant","OURS",0
315 DATA"d'innombrables petits arbustes","ARBU",4
316 DATA"de nombreuses barres de fer","BARR",6
317 DATA"*des pepites*","PEPI",5
318 DATA"une barre de fer","BARR",0
319 DATA"une chaine de fer","CHAIN",0
500 REM ********** DESCRIPTIONS DE LIEUX
501 DATA"dans la foret",0,0,0,2,0,0
502 DATA"dans la foret",0,0,1,3,0,0
503 DATA"dans la foret devant un versant de rocher",0,4,2,0,0,0
504 DATA"dans une clairiere",3,0,5,0,0,0
505 DATA"dans une clairiere",0,0,6,4,0,0
506 DATA"devant une galerie de mine.",0,0,0,5,0,0
600 REM ******************* MESSAGES
601 m$(1)="Je ne vois rien de special."
602 m$(2)="Je ne suis pas si fort !"
603 m$(3)="Comment vois-tu cela ?"
604 m$(4)="L'ours prend le miel et disparait"
605 m$(5)="dans les profondeurs de la grotte."
690 ok$="D'accord"
699 REM *********** 2eme TITRE: INTRODUCTION
700 CLS:PEN 1:PRINT"   BIENVENUE DANS LA MINIVERSION DE":PRINT:PEN 2:PRINT TAB(1
5)CHR$(34);"L'IVRESSE DE L'OR";CHR$(34):PRINT:PEN 1
705 PRINT STRING$(40,208)
710 PRINT"A la poursuite de la fortune dans le    nouveau monde, vous avez renco
ntre il y a quelques jours un vieillard agonisant auquel vous avez porte assista
nce dans  ses derniers moments."
720 PRINT"Par reconnaissance il vous a parle de savieille mine d'or et des reste
s de son  tresor qui y sont caches."
730 PRINT"Sur le chemin de cette mine vous avez   echappe a d'innombrables peril
s. Vous   aurez bientot atteint votre but et vous saurez enfin si le vieillard a
 dit vrai ou s'il delirait."
```

```
740 PRINT STRING$(40,210)
810 FOR i=1 TO av
815 READ verb$(i):verb$(i)=UPPER$(verb$(i))
820 NEXT i
830 FOR objet=1 TO ao
835 READ ob$(objet),nom$(objet),ob(objet):nom$(objet)=UPPER$(nom$(objet))
840 NEXT objet
845 FOR lieu=1 TO AR
850 READ lieu$(lieu)
855 FOR direction=1 TO 6
860 READ passage(lieu,direction)
865 NEXT direction
870 NEXT lieu
880 PRINT:INPUT"  Voulez-vous des conseils";entree$:entree$=UPPER$(entree$)
890 IF LEFT$(entree$,1)="O" THEN GOSUB 900
895 GOTO 1000
899 REM ******** 3eme TITRE: INSTRUCTIONS
900 MODE 1:PAPER 0:INK 0,0:BORDER 0:PEN 1:INK 1,25
910 PRINT"       CPC - Ventures"
920 PRINT TAB(15)CHR$(164);" 1984  by J";CHR$(178);"rg Walkowiak"
925 PRINT STRING$(40,208):PRINT"Imaginez un robot que vous pourriez     commande
r avec de nombreux ordres."
930 PRINT"Je suis ce robot et je vais m'exposer   pour vous aux aventures les pl
us        risquees."
940 PRINT"Pour que vous puissiez me diriger       utilement je vous decrirai cha
que fois  precisement la situation dans laquelle  je me trouve."
950 PRINT"Vous me direz ensuite avec deux mots    comme par exemple PRENDS COUTE
AU ce que je dois faire.":PRINT
960 PRINT"Je comprends en outre les instructions  INVENTAIRE, INSTRUCTIONS ainsi
 que END."
970 PRINT STRING$(40,210)
980 INK 3,12,24:INK 2,24,12:PEN 3: PRINT"        Appuyez";:PEN 2:PRINT" <sur une
 TOUCHE>";:PEN 1:PRINT" ..."
990 entree$=INKEY$:IF entree$="" THEN 990 ELSE CLS:MODE 1:INK 1,2:INK 2,14:INK 3
,26:RETURN
1000 MODE 1:PAPER 0:INK 0,0:BORDER 0:PEN 1:INK 1,2:PEN 2:INK 2,14:PEN 3:INK 3,26

1010 lignevide$=STRING$(39," ")
1020 DATA le Nord, le Sud, l'Ouest, l'Est, en haut, en bas
1030 FOR direction=1 TO 6
1040 READ direction$(direction)
1050 NEXT direction
1070 CLS
1080 PRINT:PRINT
1090 LOCATE 1,1
1100 FOR ligne=1 TO 10
1110 PRINT lignevide$
1120 NEXT ligne
1130 LOCATE 1,1:PEN 1
1140 PRINT"Je suis ";
1150 PRINT lieu$(joueur);:PRINT"."
1160 PRINT"Je vois ";
1170 FOR i=1 TO ao
1180 IF ob(i)<>joueur THEN 1210
1190 IF POS(#0)+LEN(ob$(i))+2<=39 THEN PRINT ob$(i);", ";:GOTO 1210
1200 IF POS(#0)+LEN(ob$(i))+2>39 THEN PRINT:GOTO 1190
1210 NEXT i
```

```
1220 PRINT CHR$(8);CHR$(8);"."
1230 PRINT lignevide$:PEN 2
1240 PRINT"Je peux aller vers ";:imprime=0
1250 FOR direction=1 TO 6
1260 IF passage(joueur,direction)=0 THEN GOTO 1310 ELSE imprime=-1
1270 IF POS(#0)=20 THEN PRINT direction$(direction);:GOTO 1300
1280 IF POS(#0)+LEN(direction$(direction))<38 THEN PRINT", ";direction$(directio
n);:GOTO 1300
1290 PRINT",":PRINT direction$(direction);:GOTO 1300
1300 NEXT direction
1310 IF imprime=0 THEN PRINT"nulle part";
1320 PRINT".":PEN 3
1330 PRINT"----------------------------------------"
1340 IF ob(12)=-1 AND ob(17)=-1 THEN GOTO 4800
1390 PEN 3:LOCATE 1,25:INPUT"Que dois-je faire";entree$:entree$=UPPER$(entree$):
PEN 2
1400 IF LEN(entree$)>2 THEN 1500
1410 IF entree$="N" AND passage(joueur,1)<>0 THEN joueur=passage(joueur,1):PRINT
"D'accord":GOTO 1080
1420 IF entree$="S" AND passage(joueur,2)<>0 THEN joueur=passage(joueur,2):PRINT
"D'accord":GOTO 1080
1430 IF entree$="O" AND passage(joueur,3)<>0 THEN joueur=passage(joueur,3):PRINT
"D'accord":GOTO 1080
1440 IF entree$="E" AND passage(joueur,4)<>0 THEN joueur=passage(joueur,4):PRINT
"D'accord":GOTO 1080
1450 IF entree$="H" AND passage(joueur,5)<>0 THEN joueur=passage(joueur,5):PRINT
"D'accord":GOTO 1080
1460 IF entree$="B" AND passage(joueur,6)<>0 THEN joueur=passage(joueur,6):PRINT
"D'accord":GOTO 1080
1470 IF LEN(entree$)<3 THEN PRINT"Aucun chemin n'y mene !":GOTO 1080
1480 IF LEN(entree$)>8 THEN GOTO 2000
1499 REM ********** DEBUT INVENTAIRE
1500 IF LEFT$(entree$,3)<>"INV" THEN GOTO 1600
1510 PRINT"Voici ce que je porte sur moi:"
1520 FOR objet=1 TO ao
1530 IF ob(objet)=-1 THEN PRINT ob$(objet)
1540 NEXT objet
1550 GOTO 1080
1560 REM ********** FIN INVENTAIRE
1599 REM ********** SAVE GAME
1600 IF LEFT$(entree$,3)<>"SAV" THEN GOTO 1700
1610 PRINT"Appuyer sur PLAY & REC ...":PRINT"Sauvegarder sous quel nom ...";STRI
NG$(4,8);:INPUT entree$
1620 IF LEN(entree$)>10 THEN PRINT"Un peu plus court S.V.P !":GOTO 1610 ELSE ent
ree$="!"+entree$+".JEU":OPENOUT entree$
1625 PRINT#9,joueur
1630 FOR objet=1 TO ao
1631 PRINT#9,ob(objet)
1632 NEXT objet
1635 FOR lieu=1 TO ar
1636 FOR direction=1 TO 6
1637 PRINT#9,passage(lieu,direction)
1638 NEXT direction
1639 NEXT lieu
1645 FOR flag=1 TO af
1646 PRINT#9,fl(flag)
1647 NEXT flag
1650 CLOSEOUT
```

```
1660 PRINT ok$
1670 GOTO 1080
1699 REM ********************* LOAD GAME
1700 IF LEFT$(entree$,3)<>"LOA" THEN GOTO 1900
1710 PRINT"Rembobiner cassette et appuyer sur PLAY":PRINT"Quel jeu faut-il charg
er ...";STRING$(4,8);:INPUT entree$
1720 IF LEN(entree$)>10 THEN PRINT"Ce n'est pas possible !":GOTO 1710 ELSE entre
e$="!"+entree$+".JEU":OPENIN entree$
1725 INPUT#9,joueur
1730 FOR objet=1 TO ao
1731 INPUT#9,ob(objet)
1732 NEXT objet
1735 FOR lieu=1 TO ar
1736 FOR direction=1 TO 6
1737 INPUT#9,passage(lieu,direction)
1738 NEXT direction
1739 NEXT lieu
1745 FOR flag=1 TO af
1746 INPUT#9,fl(flag)
1747 NEXT flag
1750 CLOSEIN
1760 PRINT ok$
1770 GOTO 1080
1900 IF LEFT$(entree$,3)<>"INS" THEN GOTO 1950
1910 GOSUB 900
1920 GOTO 1080
1950 IF LEFT$(entree$,3)<>"END" THEN GOTO 1970 ELSE CLS
1960 PRINT"Nous esperons que vous aurez plus de    succes la prochaine fois !":P
RINT:PRINT:PRINT:END
1970 IF LEFT$(entree$,3)<>"HEL" THEN GOTO 2000
1971 IF joueur=4 AND ob(10)=0 THEN PRINT"J'ai failli tomber dans une fosse":GOTO
 1080
1972 IF joueur=4 AND ob(11)<>joueur AND NOT fl(2) THEN PRINT"Il me faut quelque
chose pour briser la chaine !":GOTO 1080
1975 PRINT"D'abord voir, puis reflechir !":PRINT"Et enfin agir.":GOTO 1080
1979 REM ********************* FIN HELP
2000 longueur=LEN(entree$)
2010 FOR lettre=1 TO longueur
2020 tester$=MID$(entree$,lettre,1)
2030 IF tester$<>" "THEN NEXT lettre
2040 everb$=LEFT$(entree$,longueurmot)
2050 rl=longueur-lettre
2060 IF rl<0 THEN 2090
2070 eobjet$=RIGHT$(entree$,rl)
2080 eobjet$=LEFT$(eobjet$,longueurmot)
2090 FOR numeroverb=1 TO av
2100 IF everb$=verb$(numeroverb) THEN 2130
2110 NEXT numeroverb
2120 PRINT"Je ne comprends pas ce verbe !":GOTO 1080
2130 FOR o=1 TO ao
2140 IF eobjet$=nom$(o) THEN 2200
2150 NEXT o
2160 PRINT"Je ne connais pas cet objet !":GOTO 1080
2190 REM                exam pren pos   ouvre utilise detruis
2200 ON numeroverb GOTO 5000,6000,7000,8000,9000,10000
```

```
4500 CLS:REM                    JOUEUR MORT
4600 PRINT"Meme cela !":PRINT:PRINT m$(0)
4610 PRINT:PRINT"Je suis mort !":PRINT
4620 INPUT"Dois-je essayer encore une fois ";entree$:entree$=UPPER$(entree$)
4630 IF LEFT$(entree$,1)="O" THEN RUN
4640 GOTO 1960
4800 CLS:REM                  JOUEUR GAGNE
4810 PRINT"Toutes nos felicitations !"
4820 PRINT:PRINT"Vous avez resolu l'epreuve qui vous     etait imposee et vous p
ouvez maintenant vous lancer dans une autre aventure."
4830 PRINT:PRINT:PRINT:PRINT:PRINT:END
4999 REM ********** JOUER UN COUP
5000 IF ob(o)<>joueur AND ob(o)<>-1 THEN GOTO 5900
5002 IF o=1 THEN PRINT m$(1):GOTO 1080
5003 IF o=3 THEN PRINT m$(1):GOTO 1080
5004 IF o=4 THEN PRINT"Il y a une etagere dans un coin.":ob(8)=joueur:GOTO 1080
5005 IF o=5 THEN PRINT"La bonbonne est pleine de miel.":GOTO 1080
5006 IF o=6 THEN PRINT m$(1):GOTO 1080
5007 IF o=7 AND ob(9)=0 THEN PRINT"Il y a des explosifs dans la caisse.":GOTO 10
80
5008 IF o=8 AND ob(5)=0 THEN PRINT"Il y a une bonbonne sur l'etagere.":ob(5)=jou
eur:GOTO 1080
5009 IF o=8 AND ob(5)<>0 THEN PRINT m$(1):GOTO 1080
5010 IF o=10 THEN PRINT"Il y a un coffre d'acier dans le puits.":ob(11)=joueur:G
OTO 1080
5011 IF o=11 AND NOT fl(1) THEN PRINT"Il est ferme par une chaine de fer.":ob(19
)=joueur:GOTO 1080
5012 IF o=11 AND fl(1) AND NOT fl(2) THEN PRINT"Je ne vois rien de special
     exterieurement.":GOTO 1080
5013 IF o=11 AND fl(1) AND fl(2) AND ob(12)=0 THEN PRINT"Il est plein de pieces
d'argent.":ob(12)=joueur:GOTO 1080
5014 IF o=12 THEN PRINT"C'est exactement ce que je cherche !":GOTO 1080
5015 IF o=13 THEN PRINT"J'ai effraye un ours !":ob(14)=joueur:fl(3)=0:GOTO 1080
5017 IF o=15 THEN PRINT"Il y a un puits naturel entre les       arbustes.":ob(10
)=joueur:GOTO 1080
5018 IF o=16 THEN PRINT"Elles ont l'air tres stables.":GOTO 1080
5019 IF o=17 THEN PRINT"C'est de l'or pur !":GOTO 1080
5800 PRINT m$(1):GOTO 1080
5900 REM       objet manquant
5901 IF o=1 AND joueur=2 THEN PRINT m$(1):GOTO 1080
5902 IF o=6 AND ob(5)=-1 THEN PRINT"Il est doux et delicieux.":GOTO 1080
5903 IF o=6 AND ob(5)<>-1 THEN PRINT"Je n'ai pas de miel !":GOTO 1080
5904 IF o=9 AND (ob(7)=joueur OR ob(7)=-1) THEN PRINT"Il a l'air tres puissant !
":GOTO 1080
5905 IF o=16 AND ob(18)=-1 THEN PRINT"A peine rouillees - vraiment stables !":GO
TO 1080
5990 PRINT"Je ne vois rien de tel ici!":GOTO 1080
6000 IF ob(o)<>joueur AND ob(o)<>-1 THEN GOTO 6900
6001 IF o=1 THEN PRINT m$(2):GOTO 1080
6002 IF o=3 THEN PRINT m$(2):GOTO 1080
6003 IF o=4 THEN PRINT m$(3):GOTO 1080
6004 IF o=8 THEN PRINT m$(2):GOTO 1080
6005 IF o=11 THEN PRINT m$(2):GOTO 1080
6006 IF o=10 THEN PRINT m$(3):GOTO 1080
6007 IF o=13 THEN PRINT m$(3):GOTO 1080
6008 IF o=15 THEN PRINT m$(2):GOTO 1080
6010 IF o=5 THEN ob(5)=-1:PRINT ok$:GOTO 1080
6011 IF o=6 THEN ob(5)=-1:PRINT ok$:GOTO 1080
6012 IF o=7 THEN ob(o)=-1:PRINT ok$:GOTO 1080
6014 IF o=12 THEN ob(o)=-1:PRINT ok$:GOTO 1080
```

```
6015 IF o=14 AND joueur=5 THEN m$(0)="L'ours m'a tue.":GOTO 4500
6017 IF o=17 AND fl(3) THEN PRINT ok$:ob(o)=-1:GOTO 1080
6018 IF o=17 AND NOT fl(3) THEN m$(0)="Un ours se jette sur moi. Il m'a tue.":GO
TO 4500
6900 IF o=9 AND (ob(7)=joueur OR ob(7)=-1) THEN m$(0)="Les explosifs ont explose
 lorsque je lesai touches !":GOTO 4500
6901 IF o=16 AND ob(18)=joueur THEN PRINT ok$:ob(18)=-1:GOTO 1080
6910 IF o=1 AND joueur=2 THEN PRINT m$(2):GOTO 1080
6999 PRINT"Je ne vois rien de tel ici !":GOTO 1080
7000 IF o=16 AND ob(18)=-1 THEN PRINT ok$:ob(18)=joueur:GOTO 1080
7001 IF ob(o)<>-1 THEN PRINT"Mais je ne possede pas encore une telle chose !":GO
TO 1080
7010 IF o=6 AND joueur=5 THEN ob(6)=0:fl(3)=-1:PRINT m$(4):PRINT m$(5):ob(14)=0:
GOTO 1080
7020 IF o=5 AND joueur=5 THEN ob(5)=0:fl(3)=-1:PRINT m$(4):PRINT m$(5):ob(14)=0:
GOTO 1080
7900 ob(o)=joueur:PRINT ok$:GOTO 1080
8000 IF ob(o)<>joueur AND ob(o)<>-1 THEN PRINT"Il n'y a rien de tel ici !":GOTO
1080
8005 IF o=4 AND joueur=3 THEN PRINT"La cabane etait deja ouverte.":GOTO 1080
8010 IF o=5 THEN PRINT ok$:GOTO 1080
8020 IF o=11 AND NOT fl(1) THEN PRINT"La chaine ne le permet pas.":GOTO 1080
8999 PRINT"Je ne comprends pas ce que tu veux dire.":GOTO 1080
9000 IF o=16 AND ob(18)=-1 AND joueur=4 THEN PRINT"La chaine se brise.":fl(1)=-1
:GOTO 1080
9999 PRINT"Je ne comprends pas ce que tu veux dire.":GOTO 1080
10000 IF o=16 AND joueur=4 AND ob(18)=-1 THEN PRINT"La chaine se brise.":fl(1)=-
1:GOTO 1080
10010 IF o=16 AND joueur=4 AND ob(18)<>-1 THEN PRINT"Avec quoi ?":GOTO 1080
10999 PRINT"Je ne comprends pas ce que tu veux dire.":GOTO 1080
```

CHAPITRE 4
- PERFECTIONNEMENT -

DU JEU D'AVENTURE CORRECT AU JEU D'AVENTURE PARFAIT

Si vous avez joué avec la miniversion de 'L'ivresse de l'or' ou si vous avez conçu vous-même un jeu d'aventure sur la base des techniques acquises jusqu'ici, vous penserez comme moi que le standard de programmes courants du type jeux d'aventure a été tout à fait atteint.

Pourtant on trouve ici et là des jeux d'aventure qui offrent quelque chose de spécial. Pour pouvoir s'élever au dessus de la concurrence, leur bienveillance envers l'utilisateur a été accentuée ou, lors de la réalisation du jeu, les futilités de la vie quotidienne ont été utilisées et ne servent qu'à rendre au joueur les choses encore plus difficiles.
De plus il existe déjà des programmes qui à grands renforts de publicité attirent l'attention sur leurs particularités incomparables.
Ainsi l'ouïe du joueur est prise en considération, cela va du bip sonore qui signale une instruction incomprise ou non exécutable en passant par le bruitage d'objets visibles jusqu'à l'imitation de la voix humaine. Malheureusement, l'utilisation ludique de tous ces suppléments ne vaut rien et on devrait bien réfléchir avant de sacrifier son argent pour une course à l'effet occupant beaucoup de place en mémoire. Ne vaut-il pas mieux préférer un monde d'aventures plus grand grâce à la place en mémoire ainsi épargnée.

LA BIENVEILLANCE ENVERS L'UTILISATEUR

Elle est constamment exigée pour des applications de gestion, pourquoi ne le serait-elle pas aussi pour les jeux ?

Il y a bien effectivement des jeux d'aventure qui n'annoncent même pas au joueur par quels chemins il peut quitter un endroit dangereux. Ces jeux n'ont qu'un but: pousser le joueur au désespoir en lui annonçant après l'entrée des différentes directions: 'Tu ne peux pas aller dans cette direction.' .

S'il est bon que le joueur puisse et doive s'arracher les cheveux et là

Je suis d'accord, il me semble toutefois que ces programmes ont été écrits plutôt pour attirer les amateurs de jeux d'aventure grâce au slogan 'Nous vous garantissons que vous passerez des mois avant de connaître toutes les astuces de ce jeu !' .
Par bonheur, cette critique ne peut concerner notre système de jeu d'aventure. Cependant le fait que manque toute possibilité de sauvegarder l'état du jeu peut vite gâcher le plaisir de jouer.
Malheureusement ceci est d'autant plus vrai pour des jeux d'aventure qui nous ont demandé un effort particulier quant à leur développement et dont nous pouvons être fiers de la complexité et des pièges posés intelligemment.

Venons donc en aide au joueur qui doit de toutes façons avoir plus que seulement sept vies pour pouvoir connaître toutes les formes de trépas possibles dans un jeu d'aventure et créons les conditions techniques afin que, riche d'une expérience supplémentaire, il puisse poursuivre le jeu à partir de ce moment. S'il n'utilise pas au bon moment cette extension très utilE, il ne peut s'en prendre qu'à lui-même.

En outre nous évitons grâce à ces routines de nous faire du tort à nous-même car pendant la période de test d'un jeu il apparaîtra toujours nécessaire d'effectuer de petites modifications de programme qui malheureusement effacent également le contenu de toutes les variables.
Il ne reste qu'à repartir avec Run et répéter toutes les entrées indispensables pour rejoindre à nouveau le lieu dans lequel était apparue l'erreur et pour constater ensuite que la nouvelle version ne représente aucunement une amélioration.

SAVE GAME / LOAD GAME

Réfléchissons d'abord en quoi une scène donnée du jeu se distingue de la position de départ.

Aussitôt nous pensons au joueur qui, de façon plus ou moins tenace, a entrepris de bouger, nous pensons également aux objets qu'il a pris, posés, détruits, utilisés ou mangés, donc à des objets qui n'existent plus comme aussi à des choses qui sont apparues en cours de jeu.

Outre ces contenus de variables visibles et de ce fait peut-être moins importantes parce que contrôlables, nous devons aussi sauvegarder des modifications faites pour commander le cours du jeu, sinon il peut arriver qu'une action n'apporte pas le résultat souhaité malgré tous les

objets nécessaires.

Il suffit de penser aux flags qui peuvent ne pas être utilisés que pour le déroulement d'une action cohérente mais aussi en vue du contrôle d'événements apparemment distincts.
C'est la même chose pour notre tableau directionnel, ici aussi des modifications importantes sont chose courante comme nous le verrons plus loin.

En bref: les contenus des variables JOUEUR, OB(), FL() et PASSAGE(,) doivent être mémorisés, avant l'interruption d'un jeu, sur disquette ou sur cassette sous forme de fichier séquentiel.

Digression: Mémorisation externe des données

Outre la mémorisation interne des données où les données sont placées dans la propre mémoire de travail de l'unité centrale qui les a ainsi constamment à sa disposition, des mémoires externes sont également indispensables pour l'utilisation de données, quelle que soit leur nature. Ces mémoires externes élargissent la mémoire d'un ordinateur presqu'à souhait et servent essentiellement à l'archivage et à la préparation de quantités énormes de données. Font partie de ce type par exemple les cartes et les bandes perforées, les bandes magnétiques comme aussi notre lecteur de disquettes ou le lecteur de cassettes incorporé dans le CPC.

Mis à part des temps d'accès et des capacités différents, on peut constater aussi des différences dans la nature de la mémorisation des données.

Des appareils à mémoire comme des perforateurs ou lecteurs de bandes perforées ou des enregistreurs de données, travaillent avec des fichiers séquentiels et sont très comparables à des rouleaux de papyrus datant de l'Antiquité, tandis qu'un lecteur de disquettes travaille généralement avec des fichiers relatifs ce qui se rapproche d'une boîte à fiches.

Les deux procédés possèdent leurs avantages et leurs défauts qui résident surtout dans le besoin de place respectif et dans le temps nécessaire pour trouver une certaine donnée.
Ainsi chaque adresse d'un fichier clientèle demandera une carte propre dans un fichier relatif tandis que sur le rouleau (fichier séquentiel),

toutes les adresses peuvent être serrées les unes à la suite des autres, ce qui économise du papier. Mais si l'on cherche l'adresse du client 234, on prend la 234ème carte, car bien entendu elles sont toutes numérotées, par contre avec le rouleau on ne pourra éviter de lire toutes les adresses ou tout du moins de les compter depuis le début.

Les lecteurs qui souhaiteraient désormais s'intéresser aux fichiers doivent le faire sur la base de livres spécialisés, sinon il suffira de faire remarquer que la mémorisation séquentielle est exactement le bon procédé pour notre problème, car nous voulons mémoriser et recharger plus tard nos contenus de variables dans l'ordre, c'est-à-dire que nous n'avons pas besoin de méthodes d'accès particulières.

Save Game

Le transfert des données s'effectue comme il se doit à l'intérieur de boucles correspondantes, ce pour quoi il est cependant nécessaire d'augmenter les données caractéristiques du programme du nombre de flags utilisés car une valeur finale doit pouvoir être aussi attribuée à ces boucles.

De même nous devons entreprendre une modification de la ligne 1500 pour incorporer les nouvelles routines dans la recherche des instructions spéciales:

```
140 af=3

1500 IF LEFT$(entree$,3)<>"INV"THEN GO
TO 1600
```

Si lors de la dernière entrée il ne s'agissait pas de l'instruction inventaire, quelques sauts sont effectués jusqu'à ce que l'identité avec une des instructions brèves ait été constatée:

```
1599 REM****************** SAVE GAME
1600 IF LEFT$(entree$,3)<>"SAV" THEN GO
TO 1700
1699 REM ***************** LOAD GAME
1700 IF LEFT$(entree$,3)<>"LOA" THEN GO
TO 1900
```

Avant que le système d'exploitation de notre ordinateur puisse par la suite transférer les données, quelques informations comme la direction du transfert, le canal de transport et la destination doivent lui être d'abord données, une tâche dont se charge l'instruction Open en coopération avec l'instruction Print.

```
1620 OPENOUT "ETATDUJEU"
```

Plus rien n'empêche le transfert de données du jeu après cette préparation. La sortie des données sur cassette intervient alors sous la même forme qu'une sortie à l'écran, l'instruction Print doit seulement être réorientée par l'indication d'un numéro de fichier (#9).
Une instruction CLOSE qui termine la routine avant le retour au programme principal nous garantit contre des erreurs qui pourraient se produire ultérieurement.

```
930 FOR flag=1 TO af:fl(flag)=0:NEXT fla
g
1625 PRINT #9,joueur
1630 FOR objet=1 TO ao
1631 PRINT #9,ob(objet)
1632 NEXT objet
1635 FOR lieu=1 TO ar
1636 FOR direction=1 TO 6
1637 PRINT #1,passage(lieu.direction)
1638 NEXT direction
1639 NEXT lieu
1645 FOR flag=1 TO af
1646 PRINT #9,fl(flag)
1647 NEXT flag
1650 CLOSEOUT
1670 GOTO 1080
```

Load game

Pour reconstruire une situation donnée, il faut transmettre à nouveau les données aux variables correspondantes, un travail dont se charge une partie de programme presque identique:

```
1720 OPENIN "ETATDUJEU"
1725 INPUT #9,joueur
```

```
1730 FOR objet=1 TO ao
1731 INPUT#9,ob(objet)
1732 NEXT objet
1735 FOR lieu=1 TO ar
1736 FOR direction=1 TO 6
1737 INPUT #9,passage(lieu,direction)
1738 NEXT direction
1739 NEXT lieu
1745 FOR flag=1 TO af
1746 INPUT #9,fl(flag)
1747 NEXT flag
1750 CLOSEIN
1770 GOTO 1080
```

Pour donner à notre programme un aspect encore plus somptueux, nous le compléterons encore de quelques autres lignes. Il n'est alors absolument pas nécessaire de nous limiter à un seul fichier pour la sauvegarde, un deuxième fichier deviendra même indispensable si des membres de notre famille se prennent de passion pour les jeux d'aventure et veulent tenter leur chance avec le même jeu.

C'est pourquoi nous laissons aux joueurs le libre choix et leur permettons de choisir à leur guise un nom jusqu'à concurrence de neuf lettres. Le jeu d'aventure doit par la suite reconnaître ce fichier d'après le suffixe ADVDAT qui sert à marquer les fichiers d'état du jeu.

Il s'avère en outre opportun d'empêcher l'interruption du cours du jeu en cas d'erreur, ce qui peut se produire très facilement lorsqu'une mauvaise cassette (ou disquette) a été placée dans le lecteur.
Des sous-programmes complétés de cette façon ressembleront aux listings suivants.

```
1599 REM ********************* SAVE GAME
1600 IF LEFT$(entree$,3)<>"SAV" THEN GOT
O 1700
1610 PRINT"Appuyer sur PLAY & REC ...":P
RINT"Sauvegarder sous quel nom ...";STRI
NG$(4,8);:INPUT entree$
1620 IF LEN(entree$)>10 THEN PRINT"Un pe
u plus court S.V.P !":GOTO 1610 ELSE ent
ree$="!"+entree$+".JEU":OPENOUT entree$
1625 PRINT#9,joueur
1630 FOR objet=1 TO ao
1631 PRINT#9,ob(objet)
1632 NEXT objet
1635 FOR lieu=1 TO ar
1636 FOR direction=1 TO 6
1637 PRINT#9,passage(lieu,direction)
1638 NEXT direction
1639 NEXT lieu
1645 FOR flag=1 TO af
1646 PRINT#9,fl(flag)
1647 NEXT flag
1650 CLOSEOUT
1660 PRINT ok$
1670 GOTO 1080
```

```
1699 REM ********************* LOAD GAME
1700 IF LEFT$(entree$,3)<>"LOA" THEN GOT
O 1900
1710 PRINT"Rembobiner cassette et appuye
r sur PLAY":PRINT"Quel jeu faut-il charg
er ...";STRING$(4,8);:INPUT entree$
1720 IF LEN(entree$)>10 THEN PRINT"Ce n'
est pas possible !":GOTO 1710 ELSE entre
e$="!"+entree$+".JEU":OPENIN entree$
1725 INPUT#9,joueur
1730 FOR objet=1 TO ao
1731 INPUT#9,ob(objet)
1732 NEXT objet
1735 FOR lieu=1 TO ar
1736 FOR direction=1 TO 6
1737 INPUT#9,passage(lieu,direction)
1738 NEXT direction
1739 NEXT lieu
1745 FOR flag=1 TO af
1746 INPUT#9,fl(flag)
1747 NEXT flag
1750 CLOSEIN
1760 PRINT ok$
1770 GOTO 1080
```

LE VOCABULAIRE

Souvent le choix du mot juste dans un jeu d'aventure est une véritable torture, car finalement en tant que joueur on est contraint d'utiliser le même langage que le programmeur ce qui n'est pas toujours vraiment simple.
Des produits d'importation conçus pour le marché américain ou anglais pour des non-débutants sont impossibles à résoudre avec l'anglais scolaire, en tout cas toujours difficilement surmontables. Un dictionnaire anglais-français, pour cette raison, doit avec certains joueurs se trouver constamment à côté de l'ordinateur. Cependant cet accessoire aussi n'est plus valable dès que l'auteur a utilisé un langage familier ou une sorte de slang (argot).

Si beaucoup de jeux se heurtent à la barrière des langues, on est d'autant plus content d'apprendre qu'un jeu d'aventure en français est apparu.
Mais deux traits caractéristiques troublent aussitôt le jugement, car notre langue maternelle n'est pas aussi capable comme l'anglais d'exprimer les différentes situations en seulement deux mots.
Comment faire d'autre part pour que l'utilisateur ait une chance de trouver pour s'exprimer des mots qui puissent être compris par le programme?

SYNONYMES

Introduire en grande quantité des mots signifiant la même chose serait une première proposition pour résoudre le problème du langage. Comme on peut l'imaginer avec l'exemple de l'arbre, l'introduction conséquente de mots au sens identique peut augmenter le volume d'un programme ce qui représente une masse immense de travail supplémentaire et réduit considérablement la place disponible en mémoire.
Cependant une solution doit être proposée ici en nous servant à nouveau de l'exemple de notre miniversion, proposition qui séduit par sa brièveté et sa simplicité. Bien entendu il ne s'agit pas de lignes de programme indispensables. C'est à vous de choisir si vous attachez de l'importance à un vocabulaire vaste ou si vous voulez vous satisfaire de la proposition qui suit pour améliorer la facilité d'utilisation.

Le joueur de 'L'ivresse de l'or' a eu la première occasion de ne pas trouver aussitôt le mot juste au lieu 3 en tentant d'examiner de plus

près la cabane.
Il échouera avec 'Examine baraque', le succès arrivera seulement avec 'Examine cabane'.
Afin que le deuxième concept puisse être bien compris, nous devons d'abord étendre notre vocabulaire :

```
120 ao=20
320 DATA "-","baraque",0
```

Nous pouvons faire l'économie de la description détaillée. De même le zéro devient nécessaire comme lieu de stockage, puisqu'une autre cabane ne doit apparaître ni dans l'inventaire ni dans un lieu.
Le nom d'appel est nécessaire afin que le moteur du jeu d'aventure puisse déterminer le numéro d'appel correspondant ce qui est fait aussi ligne 2140.

Normalement l'exécution du programme se poursuit ligne 2200 avec un saut à l'exécution de l'instruction correspondante. On n'y trouve naturellement aucune condition qui serait remplie étant donné que nulle part une valeur de 20 n'est exigée pour o. Pour ne pas être obligé de compléter ces lignes, nous insérons l'astuce suivante,

```
2140 IF eobjet$=nom$(o) THEN 2170
2165 REM ********************** SYNONYMES
2170 IF o=20 THEN o=4
```

et les mêmes actions se produisent avec baraque qu'avec cabane.

Pour permettre à notre programme de pouvoir comprendre un grand nombre de verbes, le travail à effectuer est encore plus facile.
Un mot introduisant la même action est également proposé au vocabulaire du jeu et le tableau de saut est simplement complété du but du saut déjà utilisé.

Nous élargissons de la sorte 'L'ivresse de l'or' de la fonction identique à 'Examine', 'Regarde':

```
130 av=7
202 DATA reg
2200 ON numeroverbe GOTO 5000,6000,7000,8
000,9000,10000,5000
```

Vous voyez que l'on peut agrandir notre système avec des verbes synonymes de façon particulièrement simple, c'est pourquoi nous ne devons pas renoncer à ce plus.

Nous pouvons aussi faire encore plus simple de telle sorte que le joueur puisse se concentrer véritablement sur les solutions du problème et qu'en cherchant les mots justes il ne perde pas le fil conducteur.

Que penseriez-vous si un jeu d'aventure vous livrait tout son vocabulaire à la demande, en produisant une liste de tous les objets existants et des verbes possibles ?

Je comprends les verbes suivants :

examine
prends
pose
ouvre
utilise
detruis
allume
remplis
entre
supprime
consolide

Appuyer sur une touche

Une telle image devrait liquider tous les problémes de choix de mots, et le joueur ne perd ni temps ni soif de jouer par des entrées insensées.

Mais il faut reprendre ici pour la deuxième fois la croyance populaire: "Où il y a de la lumière on trouve aussi de l'ombre".

Imaginons un joueur qui ne connait pas encore 'L'ivresse de l'or'. Sans avoir beaucoup joué, s'il raisonne de façon logique, il développera instantanément, simplement au vu du vocabulaire, une stratégie qui devrait l'amener rapidement au but. Il connait aussitôt tout le vocabulaire et sait que quelque part se trouvent une bonbonne, un coffre et son avidité de l'argent et de pépites montera également aussitôt. Puis il se demande quels objets peuvent être détruits et son choix se portera

sur la bonbonne, la cabane et la chaîne.
Il s'efforce de trouver le coffre au trésor, et après avoir constaté qu'une chaîne en fer ne lui permet pas d'accéder au contenu, il tente de la détruire, ce qui sera difficile à réaliser à mains nues. Un autre VOC lui suggérera ensuite que la barre de fer est probablement l'unique objet adéquat et utile.

Mais les dangers sont également désamorcés, car il ne lui est pas dissimulé qu'à part lui un ours habite ce monde !

Ainsi le programmeur aura encore pour tâche d'évaluer exactement s'il doit livrer tout le vocabulaire du jeu ou seulement une partie.

Cela pourrait s'effectuer sans problème avec un autre en-tête qui sort un nombre de mots importants, seulement nous irions à l'encontre de notre but qui est de développer un système universel de jeux d'aventure parce que ce tableau devrait être recréé pour chaque programme.

Pour des jeux d'aventure importants avec un volume de vocabulaire correspondant, le listage du vocabulaire demeure cependant parfaitement valable, c'est pourquoi, lors du développement de notre système de jeux d'aventure, nous n'y renoncerons pas.

Le traitement de l'instruction VOC est effectué pour les raisons que l'on vient de citer par le moteur. Pour garantir une utilisation véritablement universelle de cette partie de programme, les mots destinés à la sortie doivent être conçus à partir des données du jeu considéré.

On a déjà dit rapidement que l'instruction Basic Restore place le pointeur de DATA au début du bloc de données et que le Read suivant affecte à nouveau le premier élément de la liste à une variable.

Par chance la disposition des données de notre jeu facilite un tel procédé, car sont nommés d'abord les verbes puis ensuite les objets, aucun temps d'exécution ne doit être perdu pour la lecture des données restantes.

Incorporons le sous-programme correspondant, en tenant compte de la fréquence probable d'un appel des lignes correspondantes, à la suite des routines INV, SAVE, et LOAD:

```
1700 IF LEFT$(entree$,3)<>"LOA" THEN GO
```

```
TO 1800
1800 IF LEFT$(entree$,3)<>"VOC" THEN GO
TO 1900
1805 CLS:PRINT"Je comprends les verbes
suivants:"PRINT:PRINT:RESTORE
```

Un Read venant après se comporte exactement de la même façon que la première instruction Read après le lancement du programme. Reprenons donc les lignes de programme correspondantes de la partie 1 avec la modification suivante: les données entrées ne sont pas mémorisées mais seulement sorties à l'écran:

```
1810 FOR i=1 TO av
1820 READ mot$:PRINT mot$
1830 NEXT i
```

Nous voulons ensuite donner assez de temps au joueur pour étudier la liste des verbes. Comme la même chose doit se produire aussi après la sortie des objets, nous réalisons les lignes correspondantes comme sous-programme.
Celui-ci est appelé ensuite avec GOSUB dans les emplacements du programme voulus:

```
1840 GOSUB 1890
1845 CLS:PRINT"et les objets suivants
me sont":PRINT"connus":PRINT
1850 FOR i=1 TO ao
1860 READ mot$,mot$,x:PRINT mot$
1870 NEXT i
1880 GOSUB 1890:CLS:GOTO 1080

1890 LOCATE 1,25:PRINT "Appuyer touche"
1895 entree$=INKEY$:IF entree$="" THEN
 1895 ELSE CLS:RETURN
```

Il ne faut pas oublier que sans l'utilisation d'extensions Basic correspondantes, les lignes DATA sont lues également de façon séquentielle. C'est pourquoi la ligne 1860 affecte à la variable de chaîne mot$ en premier lieu la description complète de l'objet qui, étant donné que nous n'avons pas besoin de ce long texte à cet endroit, est ensuite remplacée aussitôt par le nom d'appel.

Pour terminer le listage de tout le vocabulaire, on appelle à nouveau le sous-programme à partir de la ligne 1890 afin qu'en appuyant une autre touche le jeu puisse se poursuivre.

Les lignes précédentes rempliront leur fonction dans la mesure où moins de vingt mots devront être sortis. Si ce nombre est dépassé, le temps d'apparition des premiers mots à l'écran sera trop court pour apporter une aide valable au joueur.

Pour résoudre le problème, nous introduisons la variable de contrôle ligne. Elle est mise à zéro au début de la sortie et elle est augmentée de un à la sortie de chaque mot. Si pour vingt mots un nombre correspondant de lignes a été utilisé, nous effaçons l'écran et remettons le compteur sur un.

```
1849 ligne=0
1850 FOR i=1 TO ao
1855 ligne=ligne+1
1860 READ mot$,mot$,x:PRINT mot$
1865 IF ligne=20 TNEN GOSUB 1890
1866 IF ligne=20 THEN ligne=1:CLS
1870 NEXT i
```

Le déroulement d'un programme apporte maintenant le résultat souhaité, pourtant les abréviations EXA, OUVR et toutes les autres, ne correspondent en aucun cas au standard de nos programmes.

Pour obtenir un résultat plus satisfaisant, nous modifions les verbes et objets aux lignes DATA (200 - 500) prévues pour eux et nous donnons la peine de les écrire en entier sans nous soucier de la longueur des mots.

Par cette manipulation il s'avère nécessaire de compléter les lignes 815 et 835. En effet, de cette façon, nous ne gâcherons pas inutilement la place en mémoire prévue pour les variables.

Pourquoi le mot doit-il être mémorisé en entier si pour identifier avec précision un coup du jeu trois ou quatre lettres suffisent.

```
815 READ verbe$(i):verbe$(i)=LEFT$(verbe$(i
),longueurmot):verbe$(i)=UPPER$(verbe$(i))
835 READ ob$(objet),nom$(objet),ob
(objet):nom$(objet)=left$(nom$
```

```
(objet),longueurmot):nom$(objet)=UP
PER$(objet)
```

HELP

Le développement d'une routine Help doit être maintenant le dernier pas pour augmenter la facilité d'utilisation de notre jeu d'aventure.

Si le joueur croit qu'il s'est totalement trompé, il essaiera de recueillir des informations plus utiles grâce à la simple entrée de Help.

S'il s'agit d'un jeu simple, parce que pendant son déroulement suffisament d'indications sont données, nous nous simplifions la tâche:

```
 1959 REM ********************** HELP
1960 IF LEFT$(entree$,4)<>"HELP" THEN G
OTO 2000
1970 PRINT"Je peux répéter les règles
du jeu.":PRINT"Le souhaitez-vous ?"
1971 entree$=INKEY$:IF entree$="" THEN
1971 ELSE GOSUB 900
1975 GOTO 1080
1979 REM ******************** FIN DE HELP
```

On rencontre encore plus souvent la réponse standard suivante: 'J'examinerais tout ! - Try examining things !".
Cependant, tout du moins dans les jeux d'aventure récents, on donne des conseils qui il est vrai sont parfois peu utiles parce que le joueur ne peut les comprendre.

Car comme il s'agit d'une instruction unimot, l'aide souhaitée n'est pas associée à un objet défini, ce qui ne simplifie pas l'évaluation de la situation compte tenu des difficultés rencontrées par le joueur.

Finalement le programme ne dispose pour base de travail que du seul lieu de séjour du joueur, et il n'est alors pas surprenant que de nombreux jeux donnent dans des lieux où se pose un problème toujours la même information à l'aventurier qui ne peut la comprendre parce qu'il pense jusqu'ici à un autre problème.

Recourons à l'exemple typique d'une scène de jeu à partir du lieu 4. Le

joueur a découvert le coffre au trésor et s'évertue vainement depuis un bon moment à se débarrasser de la chaîne en fer. Dans cette situation on pourrait imaginer l'indication suivante:

```
1970 IF joueur=4 THEN PRINT"Un
levier augmente la":PRINT"force !":
GOTO 1080
```

Que pourrait faire un joueur qui demande de l'aide au lieu 4, avec le message l'incitant à penser aux lois de levier, alors qu'il a échoué pour trouver le coffre et qu'il ne peut dans ce cas avoir déjà découvert la chaîne ?

Si l'on veut gâter le joueur et dans des situations particulièrement délicates le mener progressivement à la solution, cela ne sera pas possible, comme le montre l'exemple précédent, sans la formulation volumineuse de contrôles et de conditions. Les flags et également les lieux de stockage des différents objets semblent de nouveau prédestinés pour cette tâche.

Reprenons l'exemple du coffre et essayons à nouveau cette forme de situation d'aide dosée avec précision, ce qui ne doit cependant pas signifier que cette scène doit passer pour particulièrement difficile:

```
606 m$(6)="Comment puis-je detruire la
chaine ?"
607 m$(7)="Elle empêche d'ouvrir le
coffre !"

1970 IF joueur=4 AND ob(10)=0 THEN PRIN
T"J'ai failli tomber dans un trou.":
GOTO 1080
1971 IF joueur=4 AND ob(11)=joueur AND
NOT fl(2) THEN PRINT m$(6):PRINT m$(7):G
OTO 1080
```

Souvenons-nous qu'en passant dans le lieu en question, seuls les arbustes sont tout d'abord visibles, et le trou dans la terre ne doit être découvert qu'après.
Lorsque le joueur reçoit cependant le message qu'il a failli tomber dans une fosse, il s'intéressera de plus près à cet obstacle.

Une tentative supplémentaire avec Help n'apporte à ce moment aucun avantage particulier. Ce n'est qu'une fois que le coffre est devenu un objet visible faisant partie du décor que le joueur reçoit l'indication que pour examiner le contenu de ce coffre il est indispensable de détruire la chaîne et d'ouvrir le couvercle.

On peut bien espérer ensuite qu'il pourra effectuer ces actions tout seul, dans le cas contraire il serait bon que le joueur se demande s'il ne vaut pas mieux pour lui lire un roman ou s'essayer à des jeux d'arcade.

La conclusion d'une telle structure d'aide sera formée par une ligne de sécurité comme nous l'avons déjà rencontrée lors de la programmation des autres actions, une ligne qui est toujours exécutée lorqu'aucune situation spécialement définie n'est apparue:

```
1971 ...
1972 ...
1975 PRINT"D'abord voir, puis penser et
finalement agir !":GOTO 1080
```

LA MOTIVATION DU JOUEUR

Notre proposition suivante pour améliorer les jeux d'aventure augmentera notre moteur d'aventure d'une dernière instruction unimot.

Des programmes de jeux d'aventure demandent donc quant à leur solution un certain temps, cependant lorsque l'on joue durant des jours et que l'on ne progresse absolument pas, on se demande bien un jour si l'on ne doit pas charger un autre jeu.

Pour cette raison je considère les jeux d'aventure qui fonctionnent avec une évaluation par point et indiquent au joueur à quelle distance il se trouve encore de son but final, comme foncièrement plus attrayants.

Toutefois il est utile de se demander sur quelle base l'évaluation doit être effectuée. La version courante prévoit pour chaque trésor découvert ou pour chaque tâche partielle accomplie dans un jeu d'aventure à mission un certain nombre de points. Outre l'indication 'Sur 100 points tu as 40' le pourcentage est également indiqué au joueur, ce qui simplifie

l'évaluation du score. Reprenons comme exemple l' 'Adventureland' de Scott Adams: Treize trésors donnent un nombre bancal de points, avec plus de 50 points sur 100, le joueur sait cependant qu'il a parcouru la moitié du chemin.

Une solution alternative récompensera par contre tout pas en avant et n'hésitera pas non plus en contrepartie de toute aide apportée à retirer un certain nombre de points. De la même façon chaque faute entraînant la mort sera comptabilisée en négatif de sorte que dans le cas extrême un résultat négatif peut même être obtenu. Et c'est justement cette grande marge qui incite à de nouvelles tentatives, car il devrait pourtant être possible d'obtenir plus de points que le copain qui a lui aussi atteint le but.

Si un jeu d'aventure doit être équipé de cette sorte, le travail du programmeur commence par le choix de séquences appropriées, en effet la seule découverte d'un objet ne doit pas suffir à rapporter des points. Par contre le joueur peut s'attendre à un bénéfice lorsqu'il s'est joué de dangereux monstres ou lorsqu'il a trouvé des passages secrets.

Avant de nous intéresser de façon précise à l'exécution de telles versions, nous donnons d'abord au moteur la possibilité de comprendre l'instruction 'SCORE':

```
1500 IF LEFT$(entree,3)<>"INV THEN GO
TO 1560
1559 REM ********************** SCORE
1560 IF LEFT$(entree$,3)<>"SCO" THEN GO
TO 1600
```

Puis nous introduisons, pour sauvegarder la valeur des points obtenus, la variable NOTE. Il est alors important de mettre cette variable à zèro à chaque nouveau démarrage du jeu et donc aussi lors de la mort du personnage principal.

Mais le nombre de points acquis ne représente pas une indication s'il n'est pas comparé au nombre de points maximum. C'est pourquoi:

```
170 nmax=20:REM nombre de points maximum
possible de la miniversion
171 note=0:REM jusqu'ici 0 points
```

```
1561 PRINT"sur";nmax;"points tu as";
note;"points."
1565 GOTO 1080
```

Le joueur obtient les points en saisissant les pièces d'or et d'argent. Nous devons donc compléter les lignes concernées dans notre partie d'action:

```
6014 IF o=12 THEN PRINT ok$:ob(12)=-1:
note=note+10:GOTO 1080
6017 IF o=17 AND fl(3) THEN PRINT ok$.:o
b(17)=-1:note=note+10:GOTO 1080
```

Malheureusement un joueur qui ne vise que les points peut avec l'exécution actuelle du programme obtenir des résultats aussi élevé qu'il le veut.

Souvenons-nous qu'en donnant forme à l'action 'Prends' nous étions tombé d'accord qu'avec prendre on pensait 'prendre dans la main'. Cela permettait pour la pratique du jeu, outre la prise d'objets pour les stocker dans l'inventaire, aussi d'en prendre dans l'inventaire pour exécuter une action, par exemple ouvrir une porte au moyen d'une clef.

Mais cette explication se révèle maintenant gênante. Rien n'empêche le joueur d'entrer plusieurs fois 'Prends pieces d'argent' et d'augmenter à chaque fois son compte de dix points.

Comme solution s'offrent une modification des conditions préalables (ligne 6000) ainsi que l'entrée de conditions supplémentaires dans l'action elle-même.
Nous connaissons déjà ce principe pour l'exécution de l'action 'Examine', une règle similaire devrait empêcher ici qu'un objet apparaisse constamment de façon visible dans son lieu d'origine.

```
6014 IF o=12 AND ob(o)=4 THEN PRINT ok$:
ob(12)=-1:note=note+10:GOTO 1080
6017 IF o=17 AND ob(o)=5 AND fl(3) THEN
PRINT ok$:ob(17)=-1:note=note+10:G
OTO 1080
6020 IF o=12 AND ob(o)=-1 TNEN PRINT"J'a
i deja l'argent!":GOTO 1080
```

```
6021 IF o=17 AND ob(o)=-1 THEN PRINT"Je
suis deja en possession de l'or !":GOTO
 1080
```

Par contre, si vous décidez de ne concevoir l'action 'Prendre' que comme enrichissement du joueur, vous pouvez vous limiter à durcir la condition d'entrée et à sortir une indication correspondante:

```
6000 IF ob(n)<>Joueur THEN GOTO 6900
6998 IF ob(n)=Joueur THEN PRINT"Je l
'ai deja !":GOTO 1080
```

Pour nos propres jeux, à savoir tout du moins ceux de cet ouvrage, nous n'allons faire qu'une notation d'après les trésors dont s'empare le joueur. Cependant nous ne devrons pas renoncer à une évaluation des différents coups du joueur.

Au contraire, nous progresserons d'un pas en notant chaque joueur et en lui indiquant sa moyenne.

Le nombre de coups effectués servira de base pour la notation. Un joueur expérimenté ne s'attardera pas sur l'examen de détails, de même il saura tirer profit d'indications caractéristiques.
A partir de cette valeur maximum et des points obtenu nous calculons une valeur moyenne et jugeons le joueur à l'aide de ce quotient.

```
180 coup=0:note=0

1080 PRINT:PRINT:coup=coup+1

1561 PRINT"sur";nmax;"points tu as en
";coup;"coups"
1562 PRINT note;"points !":P
RINT"Cela correspond a une moyenne de
1563 PRINT note/coup;"points !"
1565 GOTO 1080
```

Nous redéfinissons aussi le critère d'une fin de jeu victorieuse. A la place d'une condition exactement définie et fonction du jeu, nous comparons maintenant le nombre de points obtenus avec la valeur maximum possible. Si la différence est nulle, le joueur a rempli sa tâche.

Pour être rationnel, ce contrôle intervient dès le début de chaque coup, avant même que des données quelconques soient transmises au joueur.

```
1085 IF note=nmax THEN GOTO 4800
```

Dans le cas où vous utilisez également le nombre de points comme critère, n'oubliez pas de supprimer toutes les autres lignes du programme qui remplissent la même fonction (numéros de ligne de 1331 à 1389).

Avec l'introduction conséquente d'un système de points, il nous reste encore pour conclure à adapter l'en-tête de la victoire à cette construction. Les données déterminées devraient être résumées une fois de plus afin qu'une notation du joueur puisse aussi s'effectuer.

En outre il sera toutefois indispensable, lorsque le jeu d'aventure sera prêt et qu'il ne présentera plus de faute, d'effectuer un jeu complet. Pour ce jeu, vous renoncerez à toutes les entrées qui ne sont pas absolument nécessaires et déterminerez le nombre de coups au moins nécessaire pour la victoire. Calculez le rapport entre les points et les entrées nécessaires et donnez ce résultat ou une valeur légèrement inférieure comme base de référence pour un très bon jeu.
Calculez alors une échelle de résultats permettant de juger chaque performance.

Ainsi la notation d'un jeu avec la miniversion de 'L'ivresse de l'or' peut être entreprise de la façon suivante :

```
4800 CLS
4805 notefinale=note/coup
4810 PRINT"Bonne chance !"
4815 PRINT"La tâche qui vous a été
confiee a ete"
4820 PRINT"remplie."
4825 PRINT"En";coup;"coups vous
avez par coup
1830 PRINT notefinale;" points."
4835 PRINT"Vous avez ainsi un ";
```

```
4840 IF note finale<.5 THEN m$(0)="
mediocre"
4845 IF note finale>0.5 THEN m$(0)=
"passable"
4850 IF note finale>1.5 THEN m$(0)=
"tres bon"
4855 IF note finale>1.0 THEN m$(0)=
"bon"
4860 PRINT m$(0);
4865 PRINT" resultat."
4899 END
```

Un titre formé de cette manière en utilisant les signes de commande, de couleur et de graphisme, incitera maints aventuriers qui sont enfin venu à bout du jeu à rejouer.

Et ceci dans le seul but d'être désigné comme étant un bon joueur.

Ces remarques suffiraient à clore le thème 'Score' si nous ne devions faire encore une remarque technique dont l'absence peut réduire sinon à néant tous nos plans et énerverait à coup sûr le joueur participant à nos programmes.

Peut-être avez-vous déjà pensé à notre routine Save Game pour laquelle on doit rendre possible une sauvegarde et un chargement des données note et coup introduites précédemment pour que le joueur ne doive pas recommencer à chaque fois avec zéro point.

```
1627 PRINT#9,note
1628 PRINT#9,coup

1727 INPUT#9,note
1728 INPUT#9,coup
```

Toutes les propositions faites jusqu'ici dans ce chapitre allaient dans le même sens: augmenter la facilité d'utilisation des programmes.

Je crois que vous pensez comme moi que les jeux d'aventure développés selon les conceptions de cet ouvrage et qui contiennent tous les perfectionnements que nous avons proposés appartiennent sont des jeux d'aventure à texte de grande classe. Concluons donc le développement

technique par ces lignes et intéressons-nous maintenant à la question de savoir comment nous pouvons rendre les jeux encore plus attrayants et aussi plus difficiles.

Nous nous en tiendrons à notre façon habituelle de procéder, après la théorie plutôt sèche, suivent des lignes de programme qui s'appliquent à notre oeuvre de débutant 'L'ivresse de l'or'.

Les pages suivantes vous montreront en particulier comment vous pouvez dresser d'autres obstacles en travers du chemin du joueur afin qu'il soit obligé d'utiliser effectivement les instructions brèves Save et Load pour atteindre un jour son but.

Pour avoir suffisament de place pour les monstres, les trappes et tout ce que l'on peut imaginer, nous nous intéresserons à nouveau tout de suite à la géographie et développerons notre monde miniature comme base vers un jeu d'aventure plus élaboré.

Dans le cas où vous envisagez de vous amuser vous-même avec un jeu d'aventure, avant d'entamer les prochains paragraphes, vous devez entrer toutes les lignes de programme manquant encore ou mieux encore, les faire entrer par quelqu'un d'autre.

Vous trouverez le listing complet de 'L'ivresse de l'or' dans lequel vous trouverez des exemples pour toutes les propositions, dans le dernier chapitre.

ORIENTATION OU DESORIENTATION

Il a déjà été mentionné au début de cet ouvrage que la plupart des jeux d'aventure se déroulent à l'intérieur d'un monde composé d'environ quarante lieux. Pourtant on a l'impression monde encore plus grand s'il y a là la forêt gigantesque, des marécages, des galeries souterraines, un labyrinthe etc, etc, etc...

Effectivement, les régions qui apparaissent justement si vastes sont des groupes de lieux peu nombreux reliés les uns aux autres de façon si habile que le joueur ne remarque en général même pas comment il y a atterri.

L'éventail des possibilités s'étend du simple lieu qui débouche toujours

sur lui-même, jusqu'à une chaîne de lieux à la fin de laquelle le joueur se retrouve au point de départ.

Utilisons ici les lieux 1 et 2 qui sont semblables dans 'L'ivresse de l'or' pour disposer dès le début du jeu d'une aire spacieuse.
Faites déboucher le lieu 1 en direction du nord et de l'ouest sur lui-même, procédez de la même façon avec le lieu 2 et essayez de vous imaginer le joueur qui part du lieu 1 et se dirige vers l'ouest. Comment peut-il savoir qu'il se trouve toujours après chaque entrée dans la même partie de la forêt ?
S'il n'a pas la chance de débuter la partie avec l'entrée 'E', il aura besoin de toute une série d'entrées pour trouver le chemin le conduisant hors de la forêt, ce qui lui retire déjà un bon nombre de points.

501 ... 1,1,1,2,0,0
502 ... 2,2,3,1,0,0

Il sera encore plus difficile de chercher le bon chemin si la sortie sud du lieu 2 ne ramène pas le joueur au lieu 2 mais l'emmème dans le lieu 1.

501 ... 1,1,1,2,0,0
502 ... 2,1,1,3,0,0

Lancez le programme et laissez-vous surprendre par l'effet de ce petit changement.

Avec trois lieux on peut déjà de cette façon construire de petits labyrinthes si les connexions sont exécutées arbitrairement, sans référence à une boussole et s'il n'existe qu'une seule voie d'accès et de sortie.

La technique courante de connexions que nous avons aussi toujours utilisée, s'en tient aux chemins réels indiqués par les points cardinaux. Avec cela nait un monde qui permet une identification aisée des lieux et, en particulier lorsque le joueur dresse une carte adéquate, une orientation.

De petits sauts au-delà de quelques lieux rendent cependant ce concept inutilisable.
Imaginons trois lieux se trouvant les uns derrière les autres, et qui offrent la description 'Je suis dans une grande caverne'.
Un joueur arrivant par l'ouest dans la caverne 1 voudra tout d'abord,

étant donné qu'il ne voit rien de particulier, connaître rapidement ce qui l'entoure et ne modifiera probablement pas la direction qu'il a déjà prise car il est nécessaire qu'il retrouve son chemin le plus facilement possible.

Après avoir traversé les lieux 2 et 3 il parvient à nouveau dans le premier lieu, ce dont il ne peut cependant se rendre compte sans actions appropriées.

En utilisant cette technique de saut il est possible de construire sans grand effort des labyrinthes paraissant gigantesques. Six lieux suffisent amplement pour qu'il soit presque impossible que le joueur trouve le bon chemin.
Même le programmeur rencontrera assez de diffilcultés pendant la période de test du programme!

Car dresser une carte devient par avec formule de travail un problème, sauf si l'on utilise un procédé alternatif qui doit être présenté ici aussi en bref.

Une façon de procéder d'après cette méthode prévoit d'abord un labyrinthe construit à l'échelle, de préférence sur du papier quadrillé.
Lors de la programmation chaque case correspond à un lieu, ce qui représente déjà beaucoup de travail.
Il est également facile de voir que pendant son excursion le joueur prend avec chaque nouveau lieu une nouvelle case, marque les murs et non les passages.

Prenez à la place six lieux parmi lesquels vous choisissez un lieu d'entrée, un de regroupement et un de sortie. Il est important pour le lieu d'entrée qu'il ne conduise que par une seule et unique direction à l'endroit visité en dernier.
La même chose vaut pour le dernier lieu, un seul et unique passage permet l'accès au nouveau pays, tous les autres chemins doivent reconduire dans le labyrinthe.
Une fonction centrale incombe donc au lieu de regroupement. De chaque lieu du labyrinthe partent deux ou plusieurs chemins vers ce lieu, toutes les sorties ramènent cependant le joueur dans le premier ou même le deuxième lieu de ce dédale.

Pour rendre la chose encore plus compliquée, les sorties du lieu qui se trouve connecté avec le dernier lieu, sont disposées de telle manière

qu'un seul chemin conduise dans le dernier lieu, mais que tous les autres mènent dans le lieu de regroupement ou dans le premier lieu.
Il va sans dire qu'une description identique est prévue pour tous les lieux et que dans la mesure du possible les six directions sont utilisées.

Le joueur devra alors avoir presque autant de chance qu'un gagnant du loto pour pouvoir quitter le labyrinthe: il devra pour cela dénicher l'unique bon chemin parmi six partant de l'avant-dernier lieu, même chose pour le dernier lieu. S'il commet une faute, il est ramené dans un autre lieu qui lui offre à son tour six possibilités.

Essayez de vous orienter dans le labyrinthe suivant (bien que vous ayez entré vous-même les connexions juste avant, cela vous semblera difficile):

```
110 ar=14

507 DATA "A l'entree de la galerie",7,0,1,5,0,
0
519 DATA dans un couloir sinueux,8,
0,0,5,0,0
524 DATA dans un couloir sinueux,10
,7,8,8,8,8
525 DATA dans un couloir sinueux,11
8,8,13,11,8,
526 DATA dans un couloir sinueux,11
,8,10,11,11,8
527 DATA dans un couloir sinueux,9,
8,10,11,10,8
528 DATA dans un couloir sinueux,8,
10,10,11,0,0
529 DATA dans un couloir sinueux,0,
11,12,11,0,0
```

CHANGENMENTS DE LIEU

En dehors des dédales de grandeur variable à souhait, nous disposons d'autres moyens pour rendre l'orientation du joueur plus difficile.

Des transferts soudains du personnage central dans un autre lieu sont très appréciés et sont communiqués au joueur par le message 'Tout tourne autour de moi'.
Mais dans le meilleur des cas, cette information doit être considérée comme un geste amical du programmeur et non comme une obligation.

Des actions de ce type sont bien souvent déclenchées par des objets magiques dont on se sert nécessairement depuis la Lampe Merveilleuse d'Aladin en les frottant fort.
Mais des formules et des philtres magiques ne sont pas non plus sans efficacité, c'est pourquoi on ne peut que conseiller au joueur d'agir avec la plus grande prudence lorsqu'il voit tomber dans quelque situation que ce soit le mot 'magie'.

D'un point de vue de technique de programme, ces changements de lieu, lors de l'utilisation de notre système de jeu, ne pose de difficultés d'aucune sorte, car il suffit d'affecter à la variable JO le numéro du nouveau lieu.

'Frotte lampe' pourrait alors être introduit comme suit (lampe = objet 3):

```
xxxx IF o=3 AND ob(3)=-1 THEN PRINT"Le
monde tourne ...":joueur=9:GOTO 1080
```

Si par la suite la description de 9 correspond encore à celle de l'ancien lieu, l'égarement du joueur devrait demeurer inévitable après le coup suivant. Ceci est vrai en particulier si le message est supprimé ou si un autre texte est choisi:

```
xxxx IF o=3 AND ob(3)=-1 TNEN PRINT ok$;
"-rien ne se produit.":joueur=9:GOTO 108
0
```

On augmente le degré de difficulté lorsque le changement de lieu du joueur ne dépend même plus des manipulations des objets mais est régi par le seul hasard.

Dans la construction de notre programme nous pouvons ordonner que le passage dans un lieu donné soit inévitable.
Si le contenu de JOUEUR correspond au numéro de ce lieu, on décide au hasard si le joueur est transféré ailleurs ou si le joueur peut séjourner

dans le but d'exécuter d'autres actions.

Quel joueur, si après avoir traversé avec succès un labyrinthe et être entré dans un lieu apparemment anodin il a été rejeté plus d'une fois dans le dédale, courra le risque de recommencer ?

Un nouveau passage du jeu 'L'ivresse de l'or' montre comment ce handicap peut être intégré:

```
110 ar=16

519 DATA dans un couloir sinueux,8,
0,15,5,0,0
521 DATA dans une galerie,11,11,11,7,11,
0
522 DATA dans une voûte rocheuse,0,15,0,0,0,0

1340 IF joueur=15 THEN IF RND(1)>.7THE
N joueur=16
```

Le joueur est transporté dans le lieu menaçant en quittant l'avant-dernier lieu menant au labyrinthe, pour l'heure lieu 8, un couloir sinueux, en direction de l'ouest.

Il peut se retirer aussitôt de ce lieu ou il peut essayer de trouver ce que peut lui apporter sa nouvelle situation.

Si le hasard le veut - et cela nous le déterminons grâce à la valeur comparative dans la fonction RND -, il parvient dans le lieu suivant (16), sinon il se retrouve dans un lieu du labyrinthe (lieu de regroupement 11) qui ne lui offre aucun point de repère pour s'orienter.

Si la fortune ne lui a pas souri, il décidera après quelques tentatives que, s'agissant là d'un autre accès au le dédale, il évitera désormais ce lieu et ne découvrira ainsi jamais une partie du trésor.

ACCES CACHES

Ils peuvent causer pour le joueur des difficultés au moins aussi grandes. Même si dans la ligne 'Je peux aller vers ...', seules les directions est et ouest sont nommées, le monde ne doit pas pour cela s'arrêter dans la

direction du nord.

De nombreux joueurs se fieront aux données fournies et ne pénètreront jamais dans la caverne. Car du fait qu'un examen de celle-ci, comme aussi d'autres actions, sont possibles à partir de la position présente, tout sentiment d'un gigantesque monde souterrain est aussitôt dissipé.

Seule une entrée comme 'Entre caverne' conduit à de nouvelles constatations:

```
130 av=7

209 DATA entre

2200 ON numero de verbe GOTO 5000,6000,7000,8
000,9000,10000,13000

13100 IF o=13 AND joueur=5 THEN joueur
=12:PRINT ok$:GOTO 1080
```

ATTENTION: Avant de vous assurer de la fonction de ces lignes, il est ind ispensable d'entrer tous les autres lieux et de modifier aussi les connexions de ceux existant déjà (lignes 500- 700).
De la même façon la ligne 110 doit être adaptée à la nouvelle situation.

Le côté positif de cette réalisation est que l'accès n'est pas du tout camouflé à l'inverse d'une autre forme d'exécution qui est également trés populaire parmi les concepteurs de jeux d'aventure.

La tâche du joueur ne consiste plus maintenant à détecter un lieu et à pénétrer dans celui-ci grâce à des mots justement choisis, mais il doit être en mesure de développer un plan pour pouvoir construire un chemin praticable y conduisant.

Dans le cas le plus simple, il s'agit d'une porte tout d'abord fermée à clef. Une annonce est fournie par le moteur par les descriptions d'objet et les données de la ligne du tableau des directions concernant ce lieu:

```
xxxx DATA une porte,porte,1
     REM LIEU 1
xxxx DATA dans une cellule,0,0,0,0,0,0
```

Ce procédé courant prévoit alors que le joueur, lors de l'examen de la cellule, découvre la porte armée d'une imposante serrure.

'Ouvre porte' ou 'Ouvre serrure' indiquent que la clef manque, ce qui incite le joueur à examiner tester tous les autres objets occupant le lieu pour tester s'ils lui sont utiles pour ouvrir la serrure.

Dans la plupart des jeux d'aventure, la seule présence de la clef dans l'inventaire suffit, les autres jeux demandent une utilisation correcte de la clef, une particularité que nous pouvons nous expliquer maintenant grâce à notre expérience pratique.

Tandis que les uns se contentent d'un simple contrôle (ob(o)=-1), les autres programmes, plus recherchés, travaillent en plus avec des flags.

Bien sûr si la porte est d'abord ouverte, une indication correspondante apparaît dans la ligne d'en-tête, après quoi nous terminons par une manipulation de la ligne du tableau PASSAGE (,) qui correspond à ce lieu.

```
xxxx IF o=y AND ob(z)=-1 AND joueur=r T
HEN PRINT ok$:passage(r,ri)=nr:GOTO 10
80
```

```
y = numero d'objet porte
z = numero d'objet clef
r = lieu actuel
ri= direction concernee
nr= nouveau lieu a atteindre
```

La ligne ci-dessus, introduite dans la partie de programme prévue pour le traitement du verbe 'ouvre', s'assure que le joueur est en possession de la bonne clef (z) et se trouve devant la bonne porte (r). Puis les données du tableau des directions sont modifiées en conséquence.

LIMITATIONS ET COMPTEUR

Je les considère comme particulièrement appropriés pour faire se distinguer un jeu d'aventure de la masse des autres jeux. Ce sont eux qui donnent à chaque programme de jeu d'aventure la fascination de la réalité et transforment certains passages de ces jeux en de véritables exercices cérébraux. Leur domaine d'action réside moins dans une participation

active au jeu que, au contraire, dans un contrôle de fond des conditions de base dans lesquelles s'inscrit le jeu.

Il faut citer ici en premier lieu la limitation des inventaires car une limite de poids est de bon ton dans ces jeux électroniques.
C'est facile à comprendre, primo cela paraît incroyable qu'un seul homme, et nous partons du principe que ce n'est pas un superman, puisse s'encombrer d'autant d'objets qu'il veut et, secundo, le jeu peut devenir trop simple si tous les objets sont également disponibles.

D'un point de vue technique, un compteur surveille le nombre de choses que le joueur porte et si la limite de charge du personnage principal prévue par le programmeur est atteinte, le port d'autres objets est refusé.

Il faut déterminer dès avant le commencement d'une action le nombre d'objets maximum pouvant être pris:

```
185 imax=5
```

La routine d'inventaire est inchangée pour le reste, car c'est au processus de la prise d'objet que nous devons le plus prêter d'attention. A toutes les actions avec le verbe 'Prends', nous mettons maintenant pour base une condition générale supplémentaire, c'est pourquoi il faut changer la ligne 6000 une deuxième fois.
Malheureusement, cela ne nécessite pas seulement une restructuration du bloc concerné, mais anéantit aussi notre rêve d'un moteur avec toutes les fonctions et restrictions, utilisable de façon universelle.

Pour cette raison nous changeons la table de saut à la ligne 2200 et exécutons le contrôle avant le début de l'exécution même de l'instruction:

```
2200 ON numeroverbe GOTO 5000,2210,7000,8
000,9000,10000
2201 REM ********** TEST SI PLACE DANS INV
2210 nombre=0
2220 FOR i=1 TO ao
2230 IF ob(I)=-1 THEN nombre=nombre+1
2240 IF nombre=imax THEN PRINT"Non merci.
- Je suis deja assez charge !":GOTO 1080
```

```
2250 NEXT i
2260 GOTO 6000: REM    EXECUTION PRENDS
2270 REM ********** FIN INVTEST
```

Alors qu'un rôle passif est dévolu à ce compteur et que la routine est appelée uniquement dans le cadre d'une action définie exactement, d'autres contrôles de fond ne limitent pas le joueur que dans la liberté de ses actions mais peuvent aussi conduire à la fin du jeu.

Cette affirmation concerne particulièrement les lignes de programme suivantes dont l'incorporation dans notre système correspond au point sur le i.

AVAILABLE LIGHT

C'est le terme couramment utilisé pour désigner cette propriété que l'on rencontre dans certains programmes de jeux d'aventure et qui peuvent par l'utilisation de difficultés supplémentaires pousser le joueur au désespoir.

Pour les passionnés des jeux d'aventure, les obstacles habituels semblent peu suffisants et comme il est notoire que rien n'est plus difficile que la vie dans notre monde quotidien, pour éliminer cet état insupportable rien de plus ne doit être fait que d'imiter ce même monde dans une vision encore plus parfaite.

Sans 'lumière disponible' nous sommes dans l'obscurité et nous aurons du mal à voir quelque chose sans parler de la possibilité d'agir. S'ajoutant le fait que nous ne nous trouvons pas dans notre chambre à coucher où cette situation serait moins gênante mais dans un décor rocailleux, désert, sauvage et accidenté, chaque pas prend l'allure d'un risque mortel.

Cela fait bien sûr notre affaire à nous, concepteurs de jeux d'aventure, et en incorporant les prochaines lignes dans notre jeu, nous nous plaisons encore plus à imaginer les réactions de nos amis auxquels nous voulons présenter notre oeuvre tout de suite après l'avoir achevée.

Par chance chaque programme d'aventure s'accommode de l'introduction de ces obstacles. On trouve souvent de vastes cavernes ou des complexes de galeries souterraines, d'autres programmes se déroulent dans une maison; des lieux nécessitant un éclairage artificiel. Si la situation devait effectivement arriver où l'action entière se déroule au grand jour, le trajet même du soleil doit être reconstitué.
Pour pouvoir réaliser un lever ou un coucher de soleil l'utilisation d'un autre compteur suffit.

Nous déterminons tout d'abord combien de temps le jour (la nuit) doit durer en nous servant des entrées du joueur comme unité de temps.
Supposons que la nuit commence 50 coups après le lever du soleil. Cette valeur sera aussi caractéristique de certains jeux d'aventure que le nombre des points obtenus, c'est pourquoi la variable correspondante LM (quantité de lumière) est également initialisée dès le début du programme:

```
186 lm=50:lumiere=-1
lt0
```

Vos connaissances désormais très vastes sur le thème des jeux d'aventure vous permettent de considérer la variable LUMIERE comme commutateur de signal, à savoir dans le sens qu'un -1 signifie une quantité de lumière suffisante pour toutes les actions et 0 l'obscurité totale.

Les deux positions possibles de ce commutateur montrent à la routine de sortie de notre moteur si le joueur peut voir quelque chose ou s'il doit tâtonner dans le noir:

```
1131 REM ****************** PAS DE LUMIERE
1132 IF lumiere THEN 1140
1133 PRINT"Je ne sais pas exactement ou je
suis."
1134 PRINT"Il fait trop sombre pour voir q
uelque chose."
1135 PRINT: PRINT"Je ne vois plus egalemen
t les sorties.":GOTO 1330
```

Après la ligne 1132 les routines de sortie utilisées jusqu'ici sont achevées dans le cas où il y a de la lumière.

Dans le cas contraire, les lignes 1133 à 1135 informent de la façon habituelle qu'il n'y a rien à voir.
En imprimant la ligne de séparation, le programme peut suivre son cours avec les lignes usuelles.

Nous devons sinon encore nous occuper de la position correcte du commutateur LUMIERE, ce qui se produit après le déroulement du nombre de coups du jeu déterminé par LM.

Le compteur de coups ne peut être utilisé pour cette opération de comparaison parce qu'un changement jour/nuit nécessite un retour du compteur à zéro; entrons alors la variable changement de lumière (LW) :

```
186 lm=50 : lumiere=-1 : lw=0

1080 coup=coup+1 : lw=lw+1
1084 IF lw=lm THEN GOSUB 3000


2999 REM *** SOUS-PROGRAMME COMMUTATEUR DE LUMIERE
3000 IF lumiere=-1 THEN lumiere=0:GOTO 3020
3010 IF lumiere=0 THEN lumiere=-1
3020 lw=0
3030 RETURN
```

Au début d'une nouvelle phase d'entrées on contrôle si la durée (LM) prévue pour une période est atteinte. Si oui, le sous-programme commutateur est appelé et l'état de LUMIERE est changé. Il n'en faut pas plus pour exposer le joueur au passage du jour à la nuit, ce qui rend la recherche d'une lampe nécessaire.

Pour 'L'ivresse de l'or' nous ne nous contenterons pas cependant de simuler le trajet du soleil mais nous faisons dépendre cet événement de la possession d'une lampe. De ce fait nous demandons en plus au joueur d'avoir toujours assez de combustible pour la source de lumière.
Dans notre cas il s'agira d'huile qu'il peut trouver dans la grotte de l'ours. Là il en remplit la bouteille qu'il doit avoir prise auparavant, puis la lampe.
Chaque remplissage remet le compteur changement de lumière sur la valeur maximum.

Pour augmenter la difficulté, au moment où le joueur la trouve, la lanterne contiendra encore un peu d'huile. S'il pénètre dans les profondeurs de la mine sans avoir fait le plein, il est inéluctablement perdu. Un pas de trop et l'aventurier trébuche et se rompt le cou.

Comme pour le reste du territoire nous n'avons pas prévu d'obscurité, la première démarche pour réaliser cette version consiste à modifier les valeurs de départ.
Une permutation ne s'effectue toujours que lorsque le compteur changement de lumière a le même contenu que quantité de lumière.

Pour permettre cependant au joueur un nombre illimité de coups de jeu, nous fixons LM à zéro et LW à un et nous assurons de ce fait que LW ne peut jamais être égal à LM:

```
186 lm=0 : lumiere=-1 : lw=1 : l1=20
```

L1 vient en plus parce que le reste d'huile doit être moindre que les remplissages complémentaires effectués plus tard. Les autres lignes peuvent être utilisées pour le premier exemple.

Cette routine est activée en premier lieu en allumant la lampe la première fois. L'exécution de cette instruction initialisera correctement, outre la sortie de 'O.K.', aussi le compteur:

```
11030 IF o=23 AND ob(23)=-1 THEN PRINT"O
.K. - La lampe brûle.":lm=l1:lw=1:GOTO
 1080
```

Pour les vingt actions suivantes, le joueur a maintenant suffisament de lumière. En remplissant la lampe, il peut faire monter cette valeur à 50:

```
12010 IF o=23 AND ob(23)=-1 AND fl(6) TH
EN l1=50:lm=50:lw=1:PRINT ok$:GOTO 1080
```

Pour permettre de compléter le stock de combustible, que la flamme brûle ou soit éteinte, nous affectons à la variable L1 ainsi qu'à LM le nombre de coups de jeu maintenant exécutables.

Dans le cas où le joueur se rend à la surface ou pour d'autres raisons éteint la lampe, nous devons sauver le contenu actuel du réservoir:

```
210 DATA eteint
2200 ... 14000
14000 IF o=23 AND OB(o)=-1 TEHN l1=lm-lw
:PRINT ok$:GOTO 1080
```

Bien sûr ces valeurs doivent être aussi sauvées lors de la sauvegarde du jeu:

```
1625 PRINT#9,Joueur:PRINT#9,l1:PRINT#9,
lm:PRINT#9,lw

1725 INPUT#9,Joueur:INPUT#9,l1:INPUT#9,
lm:INPUT#9,lw
```

Nous nous sommes ainsi assurés que cette histoire de lumière suivra vraiment le cours qui lui est destiné.

Pour cependant épargner au joueur la surprise désagréable d'une fin irréversible (s'il devait se trouver dans les profondeurs de la terre alors que l'huile s'épuise il ne pourrait plus sauver sa peau), un avertissement, pour être fair-play, est concevable.

La ligne suivante l'informera juste avant que la lampe s'éteigne du danger qui le menace:

```
1350 IF lm-lw<15 THEN PRINT"La lumiere de
la lampe s'affaiblit."
```

J'espére que les exemples présentés vous seront utiles dans la réalisation de vos propres idées.
Ainsi, il ne vous sera certainement pas difficile de veiller dans vos programmes à ce que le joueur doive prendre à un certain moment de la nourriture afin qu'il puisse continuer de prendre des choses et exécuter des actions.

De la même façon vous pouvez, à l'aide d'un autre compteur, faire dépendre les réactions du temps écoulé.

HASARDS

Tremblements de terre déclenchant une avalanche de pierres, monstres apparaissant soudainement, les ennemis naturels du héros dans le jeu ou encore des trappes s'ouvrant à un moment inopportun sont d'autres méthodes recherchées de tout côté pour dérouter le joueur.

L'intervention de ces événements est presque toujours commandée par le hasard et ceci à double titre.

Un monstre peut habiter un lieu et arracher la tête au joueur:

```
1360 IF joueur=10 THEN m$(0)="Cette fois
je n'avais pas de miel pour l'ours.":
GOTO 4500
```

Mais le piège peut tout aussi bien ne happer le joueur qu'à certains moments et à d'autres ne touchera pas à un seul de ses cheveux:

```
370 IF joueur=20 AND RND(1)>.8 THEN m$
(0)="Le sol s'est effondré sous moi.":GOTO
4500
```

La troisième version représente une combinaison des éléments déjà présentés.
En principe on pourra aussi se baser sur une mobilité des ennemis de l'aventurier, c'est pourquoi aussi le monstre devrait recevoir un certain secteur du monde du jeu comme aire de jeu:

```
1380 IF (joueur=13 OR joueur=14 OR jou
eur=16) AND RND(1)>.8 THEN m$(0)="Revo
ilà l'ours.":GOTO 4500
```

DU JEU D'AVENTURE A TEXTE AU JEU D'AVENTURE A GRAPHISME

Les programmes de jeux d'aventure présentés dernièrement possèdent généralement également, outre les caractéristiques déjà montrées, une excellente présentation optique. Pour chaque lieu du jeu il existe une image qui informe plus ou moins bien le joueur sur des choses importantes et leur contexte.

Pour l'exécution technique du programme de ces jeux d'aventure à graphisme, le joueur a en principe trois possibilités.

Il peut tout d'abord mémoriser sur disquette un graphisme correspondant pour chaque lieu et chaque objet et le charger selon le besoin dans la mémoire de l'écran de l'ordinateur. Presque tous les jeux d'aventure à graphisme utilisent de nos jours cette possibilité parce que d'une part on peut obtenir de fabuleux résultats au moyen de digitaliseurs vidéo ou d'autres accessoires comme une table de graphisme, d'autre part parce que la somme de travail supplémentaire ne consiste qu'en un sous-programme qui doit charger, en fonction du numéro de lieu (JOUEUR), un fichier et qui ne comporte dans le meilleur cas qu'une seule ligne.
Bien entendu la patience du joueur pendant le temps de chargement devra être aussi grande que la place nécessairement utilisée sur la disquette (vous savez maintenant pourquoi maints jeux d'aventure s'étendent sur trois disquettes voire plus).

La deuxiéme méthode travaille avec un interpréteur graphique, une partie du programme du jeu d'aventure qui dispose d'une instruction line ou paint et auquel on doit fournir chaque fois les données de tous les croquis et toutes les couleurs.
Ce procédé ménage mieux la place en mémoire disponible car en général un point ne demande qu'un octet et lorsque les lignes sont exécutées en traçant des lignes, un octet suffit pour les mémoriser. Du reste le programmeur de graphisme présenté par la suite travaille selon ce principe lorsqu'il dessine une nouvelle image en appuyant sur D.

Le troisiéme procédé séduit par sa rapidité, car chaque instruction nécessaire à la construction du graphisme est directement exécutée. A condition bien sûr que le systéme d'exploitation de l'ordinateur offre des instructions correspondantes et que la mémoire de travail puisse accepter les instructions indispensables.

Dans l'évolution ultérieure de nos jeux d'aventure nous nous limiterons à

ces deux dernières méthodes car leur utilisation ne demande ni accessoires techniques ni de longs sous-programmes.

Les deux solutions demandent bien sûr d'effectuer quelques petites modifications dans notre programme de moteur car les sorties devront être maintenant faites à d'autres endroits de l'écran.

Notre prochaine démarche consiste alors à réorganiser les sorties sur écran afin que seules quelques-unes des lignes inférieures de l'écran soient utilisées.
Par chance les qualités excellentes du Basic Locomotive nous servent ici aussi de telle sorte que nous faisons l'économie de modifications compliquées de notre moteur.

Deux fenêtres, à savoir window 0 pour les sorties usuelles et window 1 pour le domaine graphique sont pleinement suffisantes; il n'est pas nécessaire de modifier des instructions print car #0 correspond à la sortie standard et ainsi toutes les sorties sont redirigées.

```
1050 WINDOW#1,1, 40,1,16
1070 WINDOW#0,1,40,17,25,:PAPER 2:INK 2,1
4:CLS#0
```

LA STRUCTURE DE L'IMAGE

Confiez la construction du graphisme à quelques sous-programmes qui sont appelés à partir du moteur.
Il faut toutefois considérer que cette routine d'impression ne doit pas être appelée après chaque entrée du joueur car pourquoi un lieu devrait-il être redessiné après une action qui s'est déroulée à l'intérieur de ce lieu?

C'est pourquoi nous introduisons une autre variable de contrôle:

```
1080 PRINT:IF ancien<>Joueur THEN GOSUB
30000
1140 PRINT"Je suis ";:ancien=Joueur
```

Lorsqu'un lieu doit être dessiné à l'écran, alors on appelle le bloc des routines de graphisme.
Puis son numéro est affecté à la variable ANCIEN. Si le joueur s'éloigne

ensuite de l'endroit de l'événement. la condition est remplie en ligne 1080 et on appelle à nouveau la ligne 30000 qui choisit la bonne routine de dessin à l'aide du numéro de lieu:

```
30000 ON joueur GOSUB 31000,32000,330
00
30010 RETURN

31000 REM LIEU 1 DESSINER
31999 RETURN

32000 REM LIEU 2 DESSINER
31999 RETURN
```

Puis le cours du programme se poursuit par la sortie des descriptions d'objets de la façon habituelle.

Si vous deviez maintenant préférer entreprendre l'élaboration du graphisme avec un interpréteur, incorporez alors à partir de la ligne 30000 les routines correspondantes du programmeur-graphique présenté plus tard.
Vous n'avez plus besoin que des deux modules "charger image" et "dessiner image", modules qui commencent respectivement en lignes 7000 ou 5000.

CHAPITRE 5
- LE SYSTEME -

Les chapitres précédents vous ont donné les éléments nécessaires pour écrire de bons jeux d'aventure.
Pourtant, bien que nous ayons développé grâce à notre système un concept qui ne rend nécessaire, pour le développement des jeux d'aventure les plus divers, que la programmation des données, des conditions et des actions du jeu considéré, une dépense de travail considérable supplémentaire est indispensable avant d'entamer la phase de test et d'effectuer un premier essai du jeu.

Une simplification du travail est amenée par ce que l'on appelle un générateur de jeu d'aventure qui, après l'entrée de toutes les données nécessaires, laisse une programme fini et exploitable sur disquette ou sur cassette.

'VENTUREFIX' travaille selon ce procédé. C'est un générateur de programmes qui produit des programmes Basic prêts à fonctionner et dont vous trouverez le listing au prochain chapitre.
Cette solution demande toujours il est vrai que le plan du jeu soit entièrement terminé avant le début de la réalisation. Car avant que le générateur puisse commencer de fonctionner, le concepteur de jeux doit dresser une liste de tous les lieux, objets, verbes etc. et il doit entrer ces données par la suite lors d'une séance astreignante.

Si pendant le test d'exécution du programme ainsi produit des fautes apparaissent, on doit toujours les supprimer de la façon habituelle.

C'est pour cette raison qu'il serait vraiment souhaitable de trouver une solution qui fixe aussitôt toute pensée de l'auteur et permette en outre un contrôle et une correction instantanés.
Ce ne peut cependant être le cas que lorsque ce n'est pas un programme fini qui est produit mais lorsqu'un fichier est développé qui contient toutes les données indispensables pour le déroulement d'un jeu.

L'évaluation de ces données est confiée alors à un interpréteur qui contient les routines indispensables à l'exécution des diverses actions.

Ce chapitre vous montrera à quoi doivent ressembler un tel interpréteur et l'éditeur qui s'y rattache.

LE FICHIER

On a déjà clairement démontré que le jeu d'aventure à proprement parler n'est représenté désormais que par un fichier.
Ce fichier doit contenir toutes les indications nécessaires à un jeu donné dans un ordre défini avec précision afin qu'il soit possible pour l'interpréteur de charger des jeux d'aventure de longueur différentes.

Si nous examinons de très près des jeux d'aventure réalisés selon nos programmes, il apparaît que ceux-ci ont déjà en grande partie une fonction d'interprétation.

Les verbes comme les objets ne sont pas utilisés directement pour exécuter une action, mais on détermine d'abord deux numéros dont la combinaison détermine avec précision quelle entrée le joueur a effectuée.

Comme condition pour ce type de travail il était nécessaire d'entrer les mots sous forme de variables, un procédé qui peut aussi utiliser, à la place des lignes Data, un fichier sur disquette (cassette).

Nous avons aussi déjà déterminé les données caractéristiques de jeu et nous les avons introduites dans le programme aux lignes 100 à 185 comme données de base du jeu.

Demandons-nous maintenant de quelles valeurs tirées de ce bloc un interpréteur a besoin. L'interpréteur ne contiendra pas tous les perfectionnements de notre système mais offrira un nombre suffisant de fonctions pour permettre à un jeu de fonctionner.

Nous trouvons d'abord quelques informations sur le nombre de lieux, objets et verbes existants.
Ces valeurs limite s'avèrent indispensables à la commande correcte des boucles d'entrée et le seront également et pour la même raison pour l'interpréteur.
De même nous devrons fixer le nombre exact de toutes les informations car elles seront dorénavant conservées sans exception dans un tableau de

variables et imprimées rapidement.

C'est la même chose pour les conditions et les actions. Dans un tableau, semblable à la table des directions PASSAGE(.), nous garderons ces données disponibles. C'est pourquoi nous prenons en plus les variables AC et BC (codes d'action et codes de condition) dans la liste des données de base.

Nous ne pouvons non plus négliger le nombre de flags car notre interpréteur doit aussi être en mesure de sauvergarder l'état du jeu.

La position de départ du joueur est la prochaine valeur importante pour le jeu que nous devons placer dans notre fichier.
Le fonctionnement du système serait certes aussi garanti si JOUEUR n'était pas initialisé dès le début du jeu, mais chaque jeu devrait commencer ensuite au lieu 1.
Au premier coup d'oeil on n'aperçoit là certes aucun désavantage mais imaginez que le joueur doive commencer le jeu après un petit changement d'action à un endroit totalement différent.

S'il n'y a pas ensuite de variable JOUEUR qui puisse être reinitialisée en conséquence, vous devez reécrire toutes les descriptions de lieux en fonction de la nouvelle disposition.

Nous pouvons renoncer d'abord à une évaluation par points. Mais afin que la fin du jeu puisse être définie, nous allons prévoir une action adéquate qui puisse évaluer les flags ou les objets de l'inventaire.

N'oublions pas aussi un complément qui n'a rien à voir avec le déroulement du jeu mais qui est cependant important.
Ainsi il est recommandé de mettre également dans le fichier des informations telles que le nom de l'auteur, le nom du jeu et surtout la date de la dernière séance de travail sur le programme, afin que vous puissiez distinguer la version 5 avec seulement trois fautes de la version 1 avec plus de dix fautes.

LA STRUCTURE DE L'EDITEUR

L'éditeur a pour fonction de faciliter à l'utilisateur l'entrée comme plus tard aussi des modifications des données. Pour cela toute une série de parties de programme différentes est indispensable dont l'appel doit

s'effectuer à l'aide d'un menu.

L'ordre dans lequel doit s'effectuer le travail sera imposé dans une certaine mesure car aussi longtemps que les lieux ne sont pas là, les objets ne peuvent non plus être placés.

De façon détaillée, le déroulement de la construction d'un jeu d'aventure avec l'éditeur pourrait ressembler à ceci:

1. ENTRER LIEUX
2. ENTRER OBJETS
3. ENTRER VERBES
4. PLACER OBJETS
5. CONNECTER LIEUX
6. ENTRER TEXTES DE MESSAGE
7. ENTRER CONDITIONS & ACTIONS

8. MEMORISER FICHIER PRET

L'utilisateur choisit chaque fonction souhaitée grâce au chiffre correspondant, c'est pourquoi nous prévoyons à la suite des instructions nécessaires à l'image ci-dessus une ligne de programme avec les buts des sauts:

```
350 ON a+1 GOTO 3000,1100,1200,1300,1400
,1500,4000,1600,5000,6000
```

LES ENTREES

Avant de commencer la construction des différents sous-programmes, il nous serait utile de nous demander comment avoir une dépense de travail aussi petite que possible et un éditeur bref.

L'utilisateur doit à chaque fois être informé des données attendues afin de pouvoir effectuer des entrées correctes. Dans ce but nous construisons de simples masques d'entrée qui contiennent les indications correspondantes.
Ces masques seront largement identiques, tout au moins pour l'entrée des lieux, des objets et des verbes, c'est pourquoi une partie de programme

commune existe pour ces tâches.

Afin que ce sous-programme puisse donner les bonnes indications, ces titres doivent être disponibles avant l'appel du sous-programme:

```
1100 CLS:t1$="ENTRER ENDROIT ":t2$=" LIE
U NR.1":t3$="JE SUIS "
1105 z=n1
1110 GOSUB 510
```

Puis le nombre des lieux existants jusqu'alors est transmis à la variable Z et le masque d'entrée/sortie est appelé (1110).

Pour les former, le curseur doit sans cesse être changé de place:

```
499 REM ****** ENTREE/SORTIE DE SOUS-PROGRAMME
500 CLS
510 LOCATE 1,1:PRINT m4$:LOCATE 1,1:PRIN
T t1$
520 z=z+1
530 LOCATE 25,1:PRINT t2$;z
540 PRINT m3$
590 LOCATE 1,25:PRINT t3$;:entree$="":L
INE INPUT entree$
6000 IF LEN(entree$)=0 THEN GOTO 200
610 IF a=2 THEN PRINT:LOCATE 1,25:INPUT"
nom d'appel ";rn$(z):PRINT
620 PRINT
640 RETURN
650 REM ****************** FIN ENTREE/SORTIE
```

Les indications correspondantes sortent d'abord. A la ligne 610 on contrôle si la fonction que l'on vient d'appeler est identique à l'entrée de l'objet.
La ligne 590 reçoit l'entrée, l'affectation à la bonne variable s'effectue à l'intérieur du sous-programme qui a appelé ce sous-programme (à l'exception du deuxième nom d'objet qui est entré directement en ligne 610).

On considère que l'utilisateur veut mettre fin à la fonction appelée lorsque à la place d'une entrée on appuie seulement sur la touche ENTER, la longueur du texte d'entrée est alors zéro (ligne 600).

L'entrée des verbes et des objets est entreprise de façon semblable à cet exemple. Seule la connexion des lieux demande une autre structure de notre programme d'édition.

CONNEXION DES LIEUX

Tout d'abord, toutes les possibilités de lieux doivent être présentées afin que l'utilisateur puisse entrer les bons numéros de lieux. Etant donné que cette fonction est également nécessaire pour le placement des objets, nous construisons un autre sous-programme (12000 - 12070).

Puis pour tous les lieux et dans l'ordre, les six directions sont prospectées et les entrées sont écrites directement dans la fonction correspondante de la table des directions:

```
1499 REM ********** CONNECTER LIEUX
1500 FOR r1=i2 TO nl
1510 CLS
1515 PRINT
1520 GOSUB 12000
1530 PRINT "lieu";r1;"mene au nord au
lieu";:INPUT du(r1,1)
1540 PRINT "lieu";r1;"mene au sud au
lieu";:INPUT du(r1,2)
1585 NEXT r1
1590 GOTO 200
1591 REM ********** FIN CONNEXIONS
```

ACTIONS & CONDITIONS

Avant de continuer par l'entrée des données principales, des conditions et des actions, nous devons développer un système de codage.

Pour conserver les données nous utilisons un autre tableau bidimensionnel où cette fois-ci les lignes sont données par les verbes et les colonnes par les objets.
Pour nous permettre à nous aussi l'interprétation des conditions nous nous décidons pour un codage par lettre. Des chiffres représenteraient certes lors du traitement un avantage du point de vue de la vitesse mais se rendre compte de l'importance d'un nombre de douze chiffres est avec

certitude plus difficile que l'interprétation d'une chaîne comme "RF3S22" lorsque des conventions ont été fixées comme on peut le voir dans la combinaison suivante:

R - OBJET ET JOUEUR RESIDENT DANS LE MEME LIEU
I - OBJET EST DANS INVENTAIRE
N - OBJET N'EST PAS DANS LE LIEU
FX- FLAG X EST MIS
GX- FLAG X EST ANNULE
SXX JOUEUR DOIT ETRE DANS LE LIEU XX

Ces conditions sont pleinement suffisantes pour permettre l'exécution des actions du joueur aux seuls moments commandés par le scénario.

Même les actions prévues correspondent en principe aux remarques de la section RECAPITULATION, chapitre 3, seules s'ajoutent les nouvelles instructions Dxy et E.
L'instruction D doit amener l'interpréteur à rendre visible une sortie du lieu actuel vers le lieu x, direction y. E termine le jeu et informe le joueur de sa victoire.

```
V    -OBJET DISPARAIT
I    -OBJET VA DANS L'INVENTAIRE
NXX  -OBJET AVEC NUMERO XX REAPPARAIT
DXY  -PASSAGE VERS LIEU X DIRECTION Y
SXX  -JOUEUR EST TRANSPOSE AU LIEU XX
FX   -FLAG X EST MIS
LX   -FLAG X EST ANNULE
MXX  -MESSAGE NR. XX EST SORTI
T    -LE PERSONNAGE DU JEU MEURT
E    -FIN DU JEU ET VICTOIRE DU JOUEUR
```

LES MESSAGES

Ils peuvent être écrits sans problème, dès l'entrée, dans les bonnes variables. D'où la brièveté de la partie de programme prévue à cet effet:

```
3999 REM ****************** MESSAGES
4000 CLS:PRINT"Entrer messages"
4010 PRINT m3$
4020 am=am+1
4030 PRINT am;:INPUT m$(am)
4040 IF LEN(m$(am))=0 THEN am=am-1:GOTO
200
4050 GOTO 4020
4060 REM ************* FIN MESSAGES
```

AM garde le contrôle du nombre de messages entrés et doit être diminué, après la fin de la fonction, par l'entrée d'une étoile.

ACCES A LA CASSETTE OU A LA DISQUETTE

Ils sont nécessaires pour mémoriser le fichier ou pour le charger à nouveau dans l'ordinateur pour un nouveau traitement.
Afin que le fichier, même si un jeu n'a pas encore été complètement

réalisé, soit déjà jouable pour l'interpréteur, des données supplémentaires doivent être entrées avant toute sauvegarde.

Avant toutes choses, ce qui est important pour l'interpréteur c'est la longueur des mots car il ne doit pas gaspiller la place disponible pour les variables plus que nécessaire en mémorisant les mots en entier. Il ne doit charger que les lettres vraiment utiles.

On a déjà dit que le lieu de départ du joueur doit se trouver dans le fichier et que le nombre de flags utilisés devient important pour l'interpréteur dès que la fonction SAVE GAME est appelée.

Après l'entrée de ces valeurs toutes les données sont déposées sous forme d'un fichier séquentiel dans l'enregisteur de données.

Sinon, le listing de l'éditeur de jeu d'aventure ne présente aucune particularité afin que, surtout avec les Remarques indiquant les différents blocs, vous puissiez comprendre le mode de fonctionnement et étendre le programme selon vos désirs.

L'INTERPRETEUR DE JEU D'AVENTURE

Comprendre l'interpréteur ne vous posera pas non plus de problèmes car il est construit comme nos programmes en trois blocs.

Dans la première partie du programme on initialise à nouveau les variables avec les données nécessaires pour le jeu.
A la place des lignes DATA avec les instructions READ qui y appartiennent, nous utilisons maintenant un fichier de données construit avec l'éditeur.

Après le choix de votre fichier, ce dernier est évalué et préparé par les lignes 60 à 210.

Le bloc de programme de la ligne 1000 jusqu'à la ligne 2160 correspond au moteur que nous avons développé. Des modifications ne sont pas nécessaires.

La façon d'exécuter chaque action est cependant complètement nouvelle.

Un seul programme, l'interpréteur, doit être en mesure de permettre le déroulement de nombreux jeux différents chaque fois les uns des autres. C'est la raison pour laquelle il n'est pas possible d'utiliser des structures rigides qui ne sont jamais adaptées qu'à une seule situation. A partir d'une série d'actions différentes on doit choisir et exécuter celles qui sont justement nécessaires jusqu'à ce que l'accumulation de leurs effets donne le même résultat.

Nous suivons cette stratégie aussi bien lors du contrôle des conditions que pour exécuter les actions.

```
2200 FOR ab=1 TO LEN (bc$(vn,o))
2210 bd$(ab)=MID$(bc$(vn,o),ab,1)
2220 NEXT ab
```

Tout d'abord la valeur finale de la boucle est déterminée, valeur finale qui dépend du nombre des lettres utilisées pour le codage d'une action définie. Puis on décompose entièrement la chaîne de caractère en différentes conditions qui sont chacune affectée à un élément de la liste BD$().

Une autre boucle contrôle maintenant l'accomplissement des différentes conditions à l'intérieur d'une autre boucle.

Lors du codage il fallait cependant fixer deux ou plusieurs paramètres pour quelques conditions. Ainsi le code S (Présence du joueur dans un lieu déterminé) exige l'indication d'un numéro de lieu et le flag adéquat doit être donné par F ou G.
Pour ces raisons nous renonçons à une boucle For/Next et choisissons à la place une structure qui nous permet d'augmenter comme nous le voulons le compteur de la boucle avec des valeurs différentes.

En plus de cette variable (X) nous utilisons encore les variables RESULTAT et OK, le résultat après chaque passage de boucle devant être logiquement vrai (-1) lorsque la condition a été remplie, et logiquement faux lorsque la condition partielle à contrôler n'est pas remplie.
La valeur de cette action est ainsi reliée logiquement par 'et' au résultat déterminé auparavant.
On s'assure ainsi qu'une action ne devienne exécutable que lorsque toutes les conditions sont remplies (car ce n'est qu'alors que ok = -1 !).

```
2300 x=0:er=0:ok=-1
```

```
2310 x=x+1:ok=(ok AND er)
```

Tout parcours de boucle commence en augmentant le compteur, ce n'est qu'après que l'on contrôle si la valeur finale dépendant effectivement du nombre de conditions est atteinte ou pas.

```
2320 IF x=ab+1 THEN 2500: REM TOUTES CON
DITIONS CONTROLEES
```

Indépendamment de ok on décide maintenant si une réaction a lieu ou si le joueur doit exécuter d'autres actions préparatoires:

```
2500 IF NOT ok THEN PRINT"Cela ne va pas pour
l'instant.":GOTO 1080
3999 REM **** TOUTES CONDITIONS REMPLIES
4000 PRINT "Okay !"
```

A l'intérieur de la structure de la boucle les conditions sont contrôlées suivant l'ordre, la ligne de programme concernée étant déterminée de la même façon que par exemple dans les routines qui traitent des instructions unimot à l'intérieur de notre programme de moteur:

```
2330 IF bd$(x)<>"R"THEN 2350
2340 IF ob(o)=joueur THEN er=-1
2345 GOTO 2310
2350 IF bd$(x)<>"I"THEN 2370
2360 IF ob(o)=-1 THEN er=-1
2365 GOTO 2310
2370 IF bd$(x)<>"N"THEN 2390
2380 IF (ob(o)<>joueur AND ob(o))<>-1 T
HEN er=-1
2385 GOTO 2310
2390 IF bd$(x)<>"S"THEN 2410
2400 r1$=bd$(x+1)+bd$(x+2):x=x+2:IF jou
eur=VAL(r1$) THEN er=-1
2405 GOTO 2310
2410 IF bd$(x)<>"F" THEN 2450
2420 IF fl(VAL(bd$(x+1))=-1 THEN 2440
2430 x=x+1:GOTO 2450
2440 er=-1:x=x+1:GOTO 2310
2450 IF bd$(x)<>"G" THEN 2310
2460 IF NOT fl(VAL(bd$(x+1))) THEN 2480
```

```
2470 x=x+1:GOTO 2310
2480 er=-1:x=x+1:GOTO 2310
2490 GOTO 2310
```

Excepté les lignes 2400 à 2470, ces lignes ne vous poseront aucun problème.
A la ligne 2400 les deux élément suivants du code de condition BD$ sont également utilisés pour déterminer le numéro de lieu, puis le compteur est actualisé et la variable de résultat est éventuellement fixée sur vrai.
Les lignes prévues pour l'évaluation des flags travaillent de la même façon.

La façon de travailler des lignes de programme exécutant une instruction est identique.
A l'intérieur de ce bloc il ne resterait qu'à indiquer les lignes 5220 à 5240 qui permettent par exemple d'ouvrir une porte. En les manipulant, les valeurs correctes s'inscrivent dans le tableau des directions de telle façon que d'une part le nouveau passage est visible (5220) mais que d'autre part un chemin de retour existe aussi lorsque le joueur s'est rendu dans le nouveau lieu (5230).

Les lignes 5500 à 5700 contiennent chacune un sous-programme court qui sert à déterminer un nombre à deux chiffres (5500), un seul chiffre (5600) ou deux chiffres (5700).
On a besoin de ces valeurs pour sortir les messages corrects, les traitements corrects des flags ainsi que pour ouvrir un passage vers un lieu déterminé dans une des six directions.

Vous trouverez aussi le listing complet de l'interpréteur dans le prochain chapitre.

CHAPITRE 6
- LA PRATIQUE DES JEUX D'AVENTURE -

Ce chapitre a moins pour objectif de vous montrer comment on fait, que de vous montrer à l'aide d'exemples de programmes complets ce que l'on peut faire.

Vous trouverez ensuite le listing complet de trois jeux d'aventure se suivent qui ont été écrits en utilisant les parties de programme et les routines développées dans cet ouvrage.

Le premier concerne 'L'ivresse de l'or', un jeu d'aventure sur lequel on a déjà dit presque trop de choses; le deuxième programme s'appelle 'Le chateau enchanté'. 'Space Mission' couronne le tout. C'est un jeu bourré d'astuces, il a été réalisé sous la forme d'un jeu d'aventure graphique.

Il est cependant dommage qu'après avoir travaillé sur cet ouvrage vous puissiez lire les programmes presque comme si on vous indiquait la solution. C'est pourquoi il est recommandé de partager de la frappe des programmes avec une autre personne.

Ceci concerne particulièrement les lignes à partir de 5000, étant donné que l'histoire du jeu est programmée ici.

A la suite des jeux vous trouverez les générateurs de programme promis, notamment le programmeur de graphisme, 'Venturefix', ainsi que les programmes du système de jeux d'aventure.

Ces programmes vous permettent de produire des jeux d'aventure commandés par le menu et d'élaborer même, avec un minimum de travail, des programmes graphiques impressionnants.
Les explications nécessaires et les indications d'utilisation sont données avant les listings.

INTRODUCTION: LES JEUX D'AVENTURE

Après avoir chargé et lancé votre programme avec RUN, les premières

descriptions et les premiers messages apparaissent après l'en-tête et d'autres indications sur le moniteur.
Vous devez distinguer la moitié supérieure de la moitié inférieure de l'écran: dans la moitié supérieure vous trouvez une description du lieu où vous vous trouvez alors, puis tous les objets visibles vous sont énumérés et dans une autre ligne les directions où vous pouvez aller vous sont données.
Par contre la moitié inférieure sert à entrer vos instructions pour le héros, et à sortir les informations, dépendant de l'exécution de vos entrées, qui vous sont destinées.

VOS ENTREES:

Vos entrées se composeront normalement et en général de deux mots, un verbe et un objet. Vérifiez tout d'abord quelles longueurs de mot le jeu considéré demande, généralement les trois ou quatre premières lettres suffisent à identifier un mot.
Veillez à utiliser des termes qui sont utilisés aussi à l'intérieur des descriptions.
Cela ne signifie toutefois pas que vous devez réagir à un message comme:

JE VOIS UNE PORTE ROUGE.

QUE DOIS-JE FAIRE:

par OUVRE PORTE ROUGE, normalement l'entrée OUVRE PORTE sera pleinement satisfaisante.

De plus quelques autres formes d'entrées vous sont permises, les instructions dites unimot qui sont moins utiles pour le déroulement du jeu que pour le service qu'elles vous rendent en simplifiant votre manière de jouer.

DEPLACEMENT DANS L'ESPACE:

En général vous pouvez toujours vous déplacer dans les directions indiquées. Pour cela il suffit d'entrer l'initiale de la direction: Nord, Sud, Est, Ouest: N, S, E, O; Haut, Bas: H, B.

Il existe pourtant des exceptions et parfois une entrée du genre ENTRE

... peut être utile.

MANIPULATION D'OBJET:

Bien sûr vous ne vous contenterez pas de visiter un monde imaginaire mais vous devez agir et résoudre des problèmes.
Cela présuppose que vous découvriez tout d'abord les objets de ce monde, puis vous pouvez les EXAMINER, PRENDRE, UTILISER etc.
Pour atteindre le but le plus rapidement possible, vous devrez agir de façon logique et réfléchie.

Supposons que vous trouviez quelque part une clef. Vous pensez 'C'est bien, mais que dois-je en faire ?'
Dans la réalité vous l'observeriez, l'examineriez et décideriez après coup si vous la laissez ou si vous la considerez comme utile ou même précieuse et la prendriez alors. Vous devez absolument exécuter ces démarches dans le jeu d'aventure.

Entrez: EXAMINE CLEF. Vous obtiendrez peut-être la réponse: C'EST LA CLEF DE MON APPARTEMENT ou: C'EST UNE CLEF DE VOITURE ou: ELLE EST EN OR MASSIF. Il va de soit que vous ne laisseriez pas cette clef mais que vous la prendriez. C'est pourquoi vous entreriez aussi: PRENDS CLEF. A l'écran apparaît comme message au moins "D'accord!" (vous le rencontrerez souvent, cela signifie que l'interpréteur du jeu a compris et exécuté votre entrée.), mais d'autres actions peuvent suivre.
Bien sûr la clef n'apparaîtra plus désormais dans la ligne d'en-tête JE VOIS, car comme vous l'avez prise, elle se trouve maintenant dans la poche de votre manteau par exemple, en tout cas plus dans le lieu.

Si vous deviez avoir accumulé de cette manière dans vos poches un tas d'objets, vous risquez de ne plus savoir ce que vous possédez et c'est pourquoi vous avez la possibilité de faire un

INVENTAIRE

Chaque fois que vous voulez savoir si vous ne transportez pas trop d'objets inutiles ou quelle est la raison pour laquelle vous avez été englouti par des sables mouvants ou quels objets seraient appropriés pour remplir une mission déterminée, entrez tout simplement INV.

La réponse apparaît aussitôt: JE PORTE AVEC MOI LES CHOSES SUIVANTES: , ainsi qu'une liste de tous ces objets.

INTERRUPTION DU JEU:

Il serait étonnant que vous puissiez résoudre un jeu d'aventure d'un trait, et bien sûr vous ne voudrez pas recommencer le jour suivant en partant du début.
C'est pourquoi presque tous ces jeux permettent de sauvegarder l'état actuel du jeu.

Si vous voulez finir provisoirement un jeu, entrez simplement SAVE. De bons programmes vous demanderont le nom sous lequel la situation présente doit être sauvegardée, des programmes plus élémentaires effectuent la sauvegarde immédiatement.

Il est également recommandé d'effectuer une sauvegarde intermédiaire: par exemple il pourrait se produire que vous vouliez escalader une paroi abrupte avec de mauvaises chaussures. Il est ici très probable que vous chuterez et vous casserez le cou; par conséquent vous devriez recommencer depuis le début et répéter toutes les entrées.
Une sauvegarde effectuée au bon moment vous aurait préservé de ce travail inintéressant !

REPRISE D'UN JEU INTERROMPU:

Vous devez bien entendu mettre tout d'abord la cassette ou la disquette adéquate, puis entrer simplement LOAD. On vous demande éventuellement sous quel nom vous avez sauvegardé l'état du jeu, ou sinon il suffit de presser la touche <RETURN> pour poursuivre le jeu à l'endroit de l'interruption.

AUTRES INDICATIONS

Lorsque l'en-tête ne vous donne pas une information normale, vous devez d'abord trouver à quel genre appartient le jeu d'aventure que vous avez chargé.
Soit vous avez une mission à accomplir, une issue à trouver ou des trésors à rassembler.

Examinez tout ce que vous voyez et adoptez toujours un comportement logique (Ainsi pour OUVRE une PORTE FERMEE vous avez d'abord besoin d'une CLEF, d'un PIED DE BICHE etc.), tout au moins aussi longtemps que vous n'êtes pas informé que la magie intervient dans le jeu. Alors vous avez le droit d'effectuer des entrées absolument folles et d'espérer parvenir très rapidement à trouver la bonne combinaison.
Faites surtout attention à des informations du genre CELA NE VA PAS POUR L'INSTANT, car elles vous expliquent que vous vous trouvez sur le bon chemin.
Mais malheureusement, toutes les conditions ne sont pas encore remplies pour passer à l'exécution de l'action (par exemple, la clef manque).

Si un verbe ou un objet devait ne pas être compris, vous en serez informé et vous devriez essayer d'employer une autre formulation pour la même action.

JE NE COMPRENDS PAS CE QUE TU VEUX DIRE exprime une dernière possibilitè. Les mots que vous avez utilisés sont connus, mais n'ont aucun sens tel que vous les avez associés. Ou bien alors l'action décrite ici n'a pas été prévue par le programmeur du jeu en cours.

TUYAUX POUR RESOUDRE UN JEU D'AVENTURE

Si vous ne savez vraiment plus comment faire pour vous dépêtrer d'une situation incorrecte, vous pouvez en premier lieu essayer de vous en tirer avec HELP.

Si cette tentative ne vous apporte pas d'informations supplémentaires, rappelez-vous tous les objets découverts jusqu'alors et examinez toutes les combinaisons possibles avec les verbes déjà connus. Peut-être tomberez-vous sur une action pleine de promesses car chaque objet doit presque à coup sûr remplir une certaine fonction; les utiliser aux seules fins de décorer un lieu serait trop fastidieux à programmer.

Dans tous les cas une liste de tous les mots compris par le programme aura son utilité; celle-ci est produite par le jeu d'aventure lui-même avec l'instruction VOCabulaire.

Une carte sur laquelle sont inscrits tous les lieux avec leurs connexions sera également utile. N'oubliez cependant pas d'y porter aussi le lieu où chaque objet a été découvert.

Si vous deviez entrer involontairement ou de façon inévitable dans un labyrinthe, la réalisation d'une telle carte n'est pas aussi facile que vous vous l'imaginez peut-être en ce moment.
Car à la place de lieux vraiment présents on a souvent cartographié trois fois la même chose.
L'unique truc qui permet d'éviter cela consiste à prélever dans chaque lieu un objet de l'inventaire et à identifier ainsi chaque endroit grâce à une marque distinctive.

Rappelons le tuyau le plus important:
Effectuez sans cesse des sauvegardes intermédiaires !

J'aimerais vous faire apprécier tout particulièrement ce jeu. On a souvent recours à des trucs pour pouvoir offrir quelques surprises à l'aventurier.
Il faut d'abord s'assurer que le joueur a bien compris quelle est sa mission, un problème que vous résoudrez malheureusement lors de la frappe du programme.

De nombreuses techniques exposées dans cet ouvrage ont été utilisées, quelques lignes de programme doivent ici servir d'exemple.
On a utilisé des flags pour permettre une parfaite maîtrise des différentes étapes avant de quitter le chateau.
Si toutes les conditions sont réunies, un passage approprié s'entr'ouvre et le monde du jeu d'aventure s'agrandit de 16 lieux.

```
1 REM Le chateau, Version 1.2
2  REM (c) 1984
3  REM by  Walkowiak
11 CLS:LOCATE 9,4:PRINT"Le chateau encha
nte"
14 LOCATE 12,20:PRINT"un jeu d'aventure
de"
15 LOCATE 14,21:PRINT"J";CHR$(178);"rg W
alkowiak"
16 LOCATE 3,25:PRINT CHR$(164);" 1984  b
y DATA BECKER, Duesseldorf"
30 PLOT 261, 231:DRAW 241, 231:DRAW 241,
 221:DRAW 221, 221:DRAW 221, 231:DRAW 20
1, 231:PLOT 120, 280:DRAW 140, 260:DRAW
140, 120:DRAW 200, 120:DRAW 200, 260:DRA
W 220, 280:DRAW 120, 280:PLOT 200, 220:D
RAW 420, 220
31 PLOT 420, 120:DRAW 420, 260:DRAW 400,
 280:DRAW 500, 280:DRAW 480, 260:DRAW 48
0, 120:DRAW 420, 120:PLOT 419, 122:DRAW
201, 122:PLOT 292, 122:DRAW 292, 146:DRA
W 300, 154:DRAW 308, 154:DRAW 316, 146:D
RAW 316, 122
32 PLOT 400, 281:PLOT 400, 281:DRAW 400,
 291:DRAW 420, 291:DRAW 420, 281:DRAW 44
0, 281:DRAW 440, 291:DRAW 460, 291:DRAW
460, 281:DRAW 480, 281:DRAW 480, 291:DRA
W 500, 291:DRAW 500, 281:PLOT 220, 281:D
RAW 220, 291
33 DRAW 200, 291:DRAW 200, 281:DRAW 180,
 281:DRAW 180, 291:DRAW 160, 291:DRAW 16
0, 281:DRAW 140, 281:DRAW 140, 291:DRAW
120, 291:DRAW 120, 281:PLOT 172, 240:DRA
W 169, 237:DRAW 169, 228:DRAW 175, 228:D
RAW 175, 237
34 DRAW 172, 240:PLOT 175, 183:DRAW 175,
 192:DRAW 172, 195:DRAW 169, 192:DRAW 16
9, 183:PLOT 169, 180:DRAW 175, 180:PLOT
448, 180:DRAW 454, 180:DRAW 454, 189:DRA
W 451, 192:DRAW 448, 189:DRAW 448, 180:P
LOT 448, 228
```

```
35 DRAW 448, 237:DRAW 451, 240:DRAW 454,
 237:DRAW 454, 228:DRAW 448, 228:PLOT 42
1, 231:DRAW 401, 231:DRAW 401, 221:DRAW
381, 221:DRAW 381, 231:DRAW 361, 231:DRA
W 361, 221:DRAW 341, 221:DRAW 341, 231:D
RAW 321, 231
36 DRAW 321, 221:DRAW 301, 221:DRAW 301,
 231:DRAW 281, 231:DRAW 281, 221:DRAW 26
1, 221:DRAW 261, 231:DRAW 241, 231:DRAW
241, 221:DRAW 221, 221:DRAW 221, 231:DRA
W 201, 231:PLOT 398, 180:DRAW 398, 189:D
RAW 395, 192
37 DRAW 392, 189:PLOT 392, 189:DRAW 392,
 180:DRAW 398, 180:PLOT 350, 180:DRAW 35
6, 180:DRAW 356, 189:DRAW 353, 192:DRAW
350, 189:DRAW 350, 180:PLOT 308, 180:DRA
W 314, 180:DRAW 314, 189:DRAW 311, 192:D
RAW 308, 189
38 DRAW 308, 180:PLOT 275, 180:DRAW 275,
 189:DRAW 272, 192:DRAW 269, 189:DRAW 26
9, 180:DRAW 275, 180:PLOT 236, 180:DRAW
236, 189:DRAW 233, 192:DRAW 230, 189:DRA
W 230, 180:DRAW 236, 180:PLOT 299, 150:D
RAW 299, 129
39 PLOT 305, 129:DRAW 305, 153:PLOT 311,
 153:DRAW 311, 129:PLOT 317, 132:DRAW 29
3, 132
40 FOR i=1 TO 8000:NEXT
150  ar= 26:ao= 45:av= 9:af= 3
160  joueur= 9
170  wl= 4
190  DIM lieu$(ar),passage(ar,6),ob$(ao)
,nom$(ao),ob(ao),verb$(av)
200  REM ************************ VERBE
N
210  DATA examine
211  DATA prends
212  DATA lire
213  DATA ouvre
214  DATA detruis
215  DATA utilise
```

```
216  DATA remplis
217  DATA pose
218  DATA reveille
300  REM ******************** OBJETS
301  DATA des etageres de livres innombr
ables,etageres,1
302  DATA une table,table,1
303  DATA mon Maitre,maitre,2
304  DATA un grand pot,pot,3
305  DATA une bouteille,bouteille,0
306  DATA la fontaine du chateau,fontain
e,4
307  DATA un livre,livre,0
308  DATA des gardiens endormis,gardiens
,5
309  DATA le palan du pont-levis,pont-le
vis,5
310  DATA des habitants du chateau,habit
ants,8
311  DATA mes invites,invites,8
312  DATA un papier,papier,0
313  DATA une table de cuisine,table,3
314  DATA un couteau de boucher,couteau,
0
315  DATA une arbalete,arbalete,5
316  DATA une clef,clef,0
317  DATA une lourde porte d'acier,porte
,6
318  DATA un pont-levis,pont-levis,7
319 DATA ,,0
320 DATA ,,0
321  DATA des arbres,arbres,12
322  DATA arbres,,13
323  DATA plusieurs etageres pleines,eta
geres,14
324 DATA ,,0
325  DATA des arbres,arbres,15
326  DATA arbres,,16
327  DATA arbres,,17
328  DATA une grande pie,pie,17
329  DATA une entree de grotte,grotte,18
330  DATA du metal dore brillant,metal,1
9
```

```
331  DATA une vieille armure,armure,0
332  DATA rien de special,,21
333  DATA un marais,marais,0
334  DATA boue,boue,23
335  DATA des oeufs de serpent,oeufs,26
336  DATA un sol trompeur,sol,23
337  DATA une eau petillante,eau,24
338  DATA une douve,douve,10
339  DATA rien de special,,20
340  DATA des toiles d'araignee,toiles,0
341  DATA une bouteille d'eau bleue,bout
eille,0
342  DATA ,,0
343  DATA une fosse a serpents,fosse,22
344  DATA des serpents,serpents,26
345  DATA un autel de pierre,autel,25
500  REM ************ DESCRIPTIONS DE L
IEUX
501  DATA dans la bibliotheque du chatea
u,0,3,0,2,0,0
502  DATA dans la salle d'etude,0,0,1,0,
0,0
503  DATA dans la cuisine,1,8,0,4,0,0
504  DATA dans la cour du chateau,0,8,3,
6,0,0
505  DATA dans l'une des tours de guet,0
,0,0,0,0,11
506  DATA dans la cour du chateau,11,0,4
,0,0,0
507  DATA devant le pont-levis,0,0,6,0,0
,0
508  DATA dans la grande salle des fetes
,4,8,3,9,0,0
509  DATA dans un appartement prive,9,9,
8,0,0,0
510  DATA sur un pont-levis,0,0,7,12,0,0
511  DATA dans la cour du chateau,0,6,0,
0,5,0
```

```
512  DATA sur un chemin forestier,12,16,
10,13,0,0
513  DATA dans la foret enchantee,21,17,
12,15,0,0
514  DATA dans une grotte sombre,0,18,14
,0,0,0
515  DATA dans la foret,15,15,15,16,0,0
516  DATA dans la foret enchantee,12,21,
15,17,0,0
517  DATA dans la foret enchantee,13,22,
16,18,0,0
518  DATA au pied d'une montagne,14,0,17
,19,0,0
519  DATA dans une grotte sombre,25,0,18
,19,0,0
520  DATA dans un chateau,20,20,20,21,0,
0
521  DATA dans la foret,16,21,20,21,0,0
522  DATA dans le marais,17,0,0,23,0,0
523  DATA dans le marais,0,0,22,24,0,0
524  DATA a une source,23,24,0,23,0,0
525  DATA dans une grotte sombre,25,19,2
5,14,14,0
526  DATA dans la fosse a serpents,0,0,0
,0,22,0
600 m$(0)="Je ne comprends pas ce que tu
 veux dire!"
601  m$(1)="Je ne vois rien de special."
602  m$(2)="Je ne suis pas si fort !"
603 m$(3)="Ils semblent dormir profondem
ent."
604 m$(4)="Cela me semble peu utile."
605 m$(6)="La pie m'a vole quelque chose
."
606 m$(7)="Il y a la une vieille armure.
"
607 m$(8)="Il tient un papier dans sa ma
in."
609 m$(9)="Il y a une cle dans une poche
 "
```

```
610 m$(10)="Une inscription dit:
           Accepte notre sacrifice et s
ois nous misericordieux."
690  ok$="D'accord !"
699  REM ********** ici eventuellement
second titre
800  FOR i=1 TO av
810  READ verb$(i):verb$(i)=LEFT$(verb$(
i),w1):verb$(i)=UPPER$(verb$(i))
820  NEXT i
830  FOR objet=1 TO ao
840  READ ob$(objet),nom$(objet),ob(obje
t):nom$(objet)=LEFT$(nom$(objet),w1):nom
$(objet)=UPPER$(nom$(objet)):NEXT objet
850  FOR lieu=1 TO ar:READ lieu$(lieu)
860  FOR direction=1 TO 6:READ passage(l
ieu,direction)
870  NEXT direction:NEXT lieu
880  PRINT:INPUT"   Voulez-vous des cons
eils";entree$:entree$=UPPER$(entree$)
890  IF LEFT$(entree$,1)="O"THEN GOSUB 9
00 ELSE GOTO 1000
895 GOTO 1000
900  MODE 1:PAPER 0:INK 0,0:BORDER 0:PEN
 1:INK 1,25
910  PRINT"      CPC - Ventures":PRINT T
AB(15)CHR$(164);" 1984  by J"CHR$(178);"
rg Walkowiak
920  PRINT STRING$(40,208):PRINT"Imagine
z un robot que vous pourriez     command
er avec de nombreux ordres."
930 PRINT"Je suis ce robot et je vais me
 soumettrepour vous aux aventures les pl
us        perilleuses."
940 PRINT"Pour que vous puissiez me comm
ander     utilement, je vous decrirai ch
aque fois la situation dans laquelle je
me trouve."
```

```
950 PRINT"Vous me direz ensuite avec deu
x mots     comme par exemple PRENDS COUTE
AU ce que je dois faire.":PRINT
960 PRINT"Je comprends en outre les inst
ructions  INVENTAIRE, SAVE, LOAD et END.
"
970  PRINT STRING$(40,210):INK 3,12,24:I
NK 2,24,12:PEN 3: PRINT"              Frap
per";:PEN 2:PRINT" <une touche>";:PEN 1:
PRINT
980  entree$=INKEY$:IF entree$="" THEN 9
80 ELSE CLS:MODE 1:INK 1,2:INK 2,14:INK
3,26:RETURN
990  MODE 1:PAPER 0:INK 0,0:BORDER 0:INK
 1,2:INK 2,14:INK 3,26:PEN 1
1000  lignevide$=STRING$(40,32)
1010  DATA le Nord,le Sud,l'Ouest,l'Est,
en haut,en bas
1020  FOR direction=1 TO 6
1030  READ direction$(direction)
1040  NEXT direction
1070  CLS
1080  PRINT:PRINT
1090  LOCATE 1,1
1100  FOR ligne=1 TO 10
1110  PRINT lignevide$;
1120  NEXT ligne
1130  LOCATE 1,1:PEN 1
1140  PRINT"Je suis ";
1150  PRINT lieu$(joueur);:PRINT"."
1160  PRINT"Je vois ";:imprime=0
1170  FOR i=1 TO ao
1180  IF ob(i)<>joueur THEN 1210
1190  IF POS(#0)+LEN(ob$(i))+2<=39 THEN
PRINT ob$(i);", ";:GOTO 1205
1200  IF POS(#0)+LEN(ob$(i))+2>39 THEN P
RINT:GOTO 1190
1205  imprime=-1
1210  NEXT i
```

```
1215  IF NOT imprime THEN PRINT"rien de
special  ";
1220  PRINT CHR$(8);CHR$(8);"."
1230  PRINT lignevide$:PEN 2
1240  PRINT"Je peux aller vers ";:imprim
e=0
1250  FOR direction=1 TO 6
1260  IF passage(joueur,direction)=0 THE
N GOTO 1300 ELSE imprime=-1
1270  IF POS(#0)=20 THEN PRINT direction
$(direction);:GOTO 1300
1280  IF POS(#0)+LEN(direction$(directio
n))<38 THEN PRINT", ";direction$(directi
on);:GOTO 1300
1290  PRINT",":PRINT direction$(directio
n);:GOTO 1300
1300  NEXT direction
1310  IF imprime=0 THEN PRINT"nulle part
";
1320  PRINT".":PEN 3
1330  PRINT STRING$(40,154)
1340 IF joueur=25 THEN GOSUB 14000
1390  PEN 3:LOCATE 1,25:INPUT"Que dois-j
e faire";entree$:entree$=UPPER$(entree$)
:PEN 2
1392 IF joueur=17 THEN GOSUB 13000
1394 IF joueur=23 THEN GOSUB 15100
1396 IF joueur=15 THEN GOSUB 15200
1400  IF LEN(entree$)>2 THEN 1500
1410  IF entree$="N" AND passage(joueur,
1)<>0 THEN joueur=passage(joueur,1):PRIN
T ok$:GOTO 1080
1420  IF entree$="S" AND passage(joueur,
2)<>0 THEN joueur=passage(joueur,2):PRIN
T ok$:GOTO 1080
1430  IF entree$="O" AND passage(joueur,
3)<>0 THEN joueur=passage(joueur,3):PRIN
T ok$:GOTO 1080
```

```
1440  IF entree$="E" AND passage(joueur,
4)<>0 THEN joueur=passage(joueur,4):PRIN
T ok$:GOTO 1080
1450  IF entree$="H" AND passage(joueur,
5)<>0 THEN joueur=passage(joueur,5):PRIN
T ok$:GOTO 1080
1460  IF entree$="B" AND passage(joueur,
6)<>0 THEN joueur=passage(joueur,6):PRIN
T ok$:GOTO 1080
1470  IF LEN(entree$)<3 THEN PRINT"Aucun
 chemin n'y mene !":GOTO 1080
1480  IF LEN(entree$)>8 THEN GOTO 2000
1499  REM ********** DEBUT INVENTA
IRE
1500  IF LEFT$(entree$,3)<>"INV" THEN GO
TO 1600
1510  PRINT"Je porte sur moi:"
1520  FOR objet=1 TO ao
1530  IF ob(objet)=-1 THEN PRINT ob$(obj
et)
1540  NEXT objet
1550  GOTO 1080
1560  REM ********** FIN INVENTAIR
E
1600  IF LEFT$(entree$,3)<>"SAV" THEN GO
TO 1700
1610  PRINT"Appuyer sur REC & PLAY ...
            Sauvegarder sous quel nom ..
.";STRING$(4,8);:INPUT entree$
1620  IF LEN(entree$)>10 THEN PRINT"Plus
 court S.V.P !":GOTO 1610 ELSE entree$="
!"+entree$+".jeu":OPENOUT entree$
1630  PRINT#9,joueur
1640  FOR objet=1 TO ao
1650  PRINT#9,ob(objet)
1660  NEXT objet
```

```
1670  FOR lieu=1 TO ar:FOR direction=1 T
O 6:PRINT#9,passage(lieu,direction):NEXT
 direction:NEXT lieu
1680  FOR flag=1 TO af:PRINT#9,fl(flag):
NEXT flag
1690  CLOSEOUT:PRINT ok$:GOTO 1080
1700  IF LEFT$(entree$,3)<>"LOA" THEN GO
TO 1900
1710  PRINT"Rembobiner cassette et appuy
er sur PLAY Quel jeu faut-il charger....
.....";STRING$(10,8);:INPUT entre
e$
1720  IF LEN(entree$)>10 THEN PRINT"Ce n
'est pas possible !":GOTO 1710 ELSE entr
ee$="!"+entree$+".jeu":OPENIN entree$
1730  INPUT#9,joueur
1740  FOR objet=1 TO ao:INPUT#9,ob(objet
):NEXT objet
1750  FOR lieu=1 TO ar:FOR direction=1 T
O 6:INPUT#9,passage(lieu,direction):NEXT
 direction:NEXT lieu
1760  FOR flag=1 TO af:INPUT#9,fl(flag):
NEXT flag
1770  CLOSEIN:PRINT ok$:GOTO 1080
1900  IF LEFT$(entree$,3)<>"INS" THEN GO
TO 1950
1910  GOSUB 900:GOTO 1080
1950  IF LEFT$(entree$,3)<>"END" THEN GO
TO 2000
1960  PRINT"Nous vous souhaitons plus de
 succes la  prochaine fois.":PRINT:PRINT
:PRINT:END
2000  longueur=LEN(entree$)
2010  FOR lettre=1 TO longueur
2020  teste$=MID$(entree$,lettre,1)
2030  IF teste$<>" "THEN NEXT lettre
2040  everb$=LEFT$(entree$,wl)
2050  rl=longueur-lettre
```

```
2060  IF r1<0 THEN 2090
2070  eobjet$=RIGHT$(entree$,r1)
2080  eobjet$=LEFT$(eobjet$,w1)
2090  FOR verbnumber=1 TO av
2100  IF everb$=verb$(verbnumber) THEN 2
130
2110  NEXT verbnumber
2120  PRINT"Je ne comprends pas ce verbe
.":GOTO 1080
2130  FOR o=1 TO ao
2140  IF eobjet$=nom$(o) THEN 2200
2150  NEXT o
2160  PRINT"Je ne connais pas cet objet.
":GOTO 1080
2200  ON verbnumber GOTO 5000, 6000, 700
0, 8000, 9000, 10000, 11000, 12000, 1600
0
4500  CLS:REM JOUEUR MORT
4510  PRINT"Meme cela !":PRINT:PRINT m$(
0)
4520  PRINT:PRINT"Je suis mort !":PRINT
4530  INPUT"Dois-je essayer encore une f
ois ";entree$:entree$=UPPER$(entree$)
4540  IF LEFT$(entree$,1)="O" THEN RUN
4550  GOTO 1960
4800  CLS:REM JOUEUR GAGNE
4810  PRINT"Toutes nos felicitations !"
4820  PRINT:PRINT"Vous avez resolu la ta
che qui vous etait":PRINT"confiee et vou
s pouvez maintenant tenter":PRINT"une au
tre aventure."
4830  PRINT:PRINT:PRINT:PRINT:END
5000 IF ob(o)<>joueur AND ob(o)<>-1 THEN
 GOTO 5900
5001 IF o=16 AND joueur=20 AND ob(31)<>-
1 AND ob(40)<>-1 THEN PRINT m$(7):ob(31)
=20:ob(40)=20:GOTO 1080
5002 IF o=10 AND joueur=8 THEN PRINT m$(
3):GOTO 1080
```

```
5003 IF o=11 AND joueur=7 THEN PRINT m$(
3):GOTO 1080
5004 IF o=4 AND (joueur=3 OR ob(4)=-1) T
HEN PRINT m$(1):GOTO 1080
5005 IF o=5 AND (joueur=3 OR ob(o)=-1) T
HEN PRINT m$(1):GOTO 1080
5006 IF o=1 AND joueur=1 THEN PRINT m$(1
):GOTO 1080
5008 IF o=3 AND joueur=2 AND ob(12)=0 TH
EN PRINT m$(1):GOTO 1080
5009 IF o=3 AND joueur=2 THEN PRINT m$(1
):GOTO 1080
5010 IF o=6 AND ob(5)=0 THEN PRINT"Une b
outeille flotte sur l'eau.":ob(5)=joueur
:GOTO 1080
5011 IF o=7 AND (joueur=1 OR ob(o)=-1) T
HEN PRINT"C'est un livre sur les potions
 magiques.":GOTO 1080
5012 IF o=8 AND joueur=5 AND ob(16)=0 TH
EN PRINT m$(9):ob(16)=joueur:GOTO 1080
5013 IF o=9 AND NOT fl(1) THEN PRINT m$(
1):GOTO 1080
5014 IF o=9 AND fl(1) AND joueur=5 THEN
PRINT"Il est hors d'usage.":GOTO 1080
5015 IF o=12 AND ob(12)=-1 THEN PRINT"Il
 y a une inscription dessus.":ob(16)=jou
eur:GOTO 1080
5016 IF o=12 AND ob(12)=2 THEN PRINT"Pou
r cela, il faut d'abord que je le    pre
nne":GOTO 1080
5017 IF o=13 AND ob(14)=0 THEN PRINT"Il
y a un couteau sur la table.":ob(14)=3:G
OTO 1080
5018 IF o=2 AND (ob(7)=1 OR ob(7)=0) THE
N PRINT"Il y a un vieux livre dessus.":o
b(7)=1:GOTO 1080
5019 IF o=14 AND (ob(14)=joueur OR ob(14
)=-1) THEN PRINT"La sonnette est tres ai
gue.":GOTO 1080
```

```
5020 IF o=15 AND (ob(o)=joueur OR ob(o)=
-1) THEN PRINT m$(1):GOTO 1080
5021 IF o=16 AND (ob(16)=-1 OR ob(16)=jo
ueur) THEN PRINT m$(1):GOTO 1080
5022 IF o=17 AND joueur=6 AND fl(3)<>-1
THEN PRINT"Elle est fermee.":GOTO 1080
5023 IF o=17 AND joueur=6 AND fl(3)=-1 T
HEN PRINT"Elle est grande ouverte.":GOTO
 1080
5024 IF o=18 AND joueur=7 AND fl(1) THEN
 PRINT"Le pont-levis s'abaisse.":passage
(7,4)=10:GOTO 1080
5025 IF o=18 AND joueur=7 AND NOT fl(1)
THEN PRINT"Le pont-levis est releve.":GO
TO 1080
5026 IF o=21 THEN PRINT m$(1):GOTO 1080
5027 IF o=23 AND joueur=14 THEN m0$="C'e
tait une cuisine ensorcelee.          Le
sorcier est venu et m'a transforme enpie
rre.":GOTO 4500
5028 IF o=28 AND joueur=17 THEN PRINT"C'
est vraiment un exemplaire magnifique.":
GOSUB 13000:GOTO 1080
5029 IF o=29 AND joueur=18 THEN PRINT"Un
e odeur curieuse se degage.";:GOTO 1080
5030 IF o=30 AND (joueur=19 OR ob(30)=-1
) THEN PRINT"C'est du metal sans valeur.
":GOTO 1080
5031 IF o=31 AND (joueur=20 OR ob(31)=-1
) THEN PRINT m$(1):GOTO 1080
5032 IF o=35 AND (joueur=26 OR ob(35)=-1
) THEN PRINT m$(1):GOTO 1080
5033 IF o=36 AND joueur=23 THEN PRINT"Il
 n'a pas l'air tres solide.":GOTO 1080
5034 IF o=38 AND joueur=10 THEN PRINT m$
(1):GOTO 1080
5035 IF o=37 AND joueur=24 THEN PRINT"C'
est la fameuse eau bleue.":GOTO 1080
```

```
5036 IF o=40 AND (joueur=20 OR ob(40)=-1
) THEN PRINT m$(1):GOTO 1080
5037 IF o=43 THEN joueur=26:PRINT"D'acco
rd !":GOTO 1080
5038 IF o=45 THEN PRINT m$(10):GOTO 1080
5899 PRINT m$(1):GOTO 1080
5900 IF o=21 THEN PRINT m$(1):GOTO 1080
5901 IF o=16 AND joueur=20 AND ob(31)<>-
1 AND ob(40)<>-1 THEN PRINT m$(7):ob(31)
=20:ob(40)=20:GOTO 1080
5902 IF o=33 AND joueur=23 THEN GOSUB 15
100:PRINT"On dirait qu'il y a une source
 a l'est.":GOTO 1080
5903 IF o=16 AND (joueur=23 OR ob(34)=-1
) THEN PRINT"C'est une boue qui guerit."
:GOSUB 15100:GOTO 1080
5904 IF o=2 AND joueur=3 AND ob(14)=0 TH
EN PRINT"Sur la table il y a un couteau.
":ob(14)=3:GOTO 1080
5905 IF o=2 AND joueur=3 AND ob(14)<>0 T
HEN PRINT m$(1):GOTO 1080
5990 PRINT"Je ne vois rien de tel ici!":
GOTO 1080
6000 IF ob(o)<>joueur AND ob(o)<>-1 THEN
 GOTO 6900
6001 IF o=1 THEN PRINT m$(2):GOTO 1080
6002 IF o=6 THEN PRINT m$(2):GOTO 1080
6003 IF o=17 THEN PRINT m$(2):GOTO 1080
6004 IF o=2 THEN PRINT m$(4):GOTO 1080
6005 IF o=3 THEN PRINT m$(4):GOTO 1080
6006 IF o=8 THEN PRINT m$(4):GOTO 1080
6007 IF o=10 THEN PRINT m$(4):GOTO 1080
6008 IF o=11 THEN PRINT m$(4):GOTO 1080
6009 za=0:FOR i=1 TO ao:IF ob(i)=-1 THEN
 za=za+1:IF za=4 THEN GOTO 6999
6010 NEXT i
6011 IF o=5 THEN ob(o)=-1:PRINT ok$:GOTO
 1080
6012 IF o=7 THEN ob(o)=-1:PRINT ok$:GOTO
 1080
```

```
6013 IF o=12 THEN ob(o)=-1:PRINT ok$:GOT
O 1080
6014 IF o=15 THEN ob(o)=-1:PRINT ok$:GOT
O 1080
6015 IF o=16 THEN ob(o)=-1:PRINT ok$:GOT
O 1080
6016 IF o=14 THEN ob(o)=-1:PRINT ok$:GOT
O 1080
6017 IF o=9 THEN PRINT m$(2):GOTO 1080
6018 IF o=23 THEN PRINT m$(2):GOTO 1080
6019 IF o=28 THEN PRINT"Avec quoi dois-j
e l'attraper ?":GOSUB 13000:GOTO 1080
6020 IF o=18 THEN PRINT m$(0):GOTO 1080
6021 IF o=30 THEN ob(o)=-1:PRINT ok$:GOT
O 1080
6022 IF o=31 THEN ob(o)=-1:PRINT"J'ai mi
s l'armure.":GOTO 1080
6023 IF o=35 AND ob(31)=-1 THEN PRINT ok
$:ob(35)=-1:GOTO 1080
6024 IF o=35 AND ob(31)<>-1 THEN m$(0)="
Un serpent m'a mordu.":GOTO 4500
6025 IF o=36 THEN PRINT m$(4):GOTO 1080
6026 IF o=37 AND ob(5)=-1 THEN PRINT ok$
:ob(5)=0:ob(41)=-1:GOTO 1080
6027 IF o=37 AND ob(5)<>-1 THEN PRINT"Il
 me faut d'abord une bouteille vide.":GO
TO 1080
6028 IF o=38 THEN PRINT m$(0):GOTO 1080
6029 IF o=40 THEN ob(40)=-1:PRINT ok$:GO
TO 1080
6032 IF o=13 THEN PRINT m$(4):GOTO 1080
6033 IF o=4 THEN ob(o)=-1:PRINT ok$:GOTO
 1080
6900 IF o=16 AND joueur=23 AND ob(4)<>-1
 THEN PRINT"Il me faut d'abord un recipi
ent !":GOTO 1080
6901 IF o=16 AND joueur=23 AND ob(4)=-1
THEN PRINT ok$:ob(34)=-1:GOTO 1080
6902 IF o=2 AND joueur=3 THEN PRINT m$(4
):GOTO 1080
6903 IF o=5 AND ob(41)= JOUEUR THEN PRINT
ok$: ob(41) =-1 :GOTO 1080
```

```
6998 PRINT"Je ne vois rien de tel ici!":
GOTO 1080
6999 PRINT"Je regrette mais je porte dej
a assez de":PRINT"choses !":GOTO 1080
7000 IF o=13 AND ob(12)<>-1 THEN PRINT"J
e n'ai pas de papier.":GOTO 1080
7001 IF o=12 AND ob(o)=-1 THEN co=INT(RN
D(1)*100):PRINT"Je vois ecrit d'une ecri
ture tremblante le nombre:":PRINT co:GOT
O 1080
7002 IF o=7 AND ob(7)=-1 THEN GOTO 7650
7003 IF o=7 AND ob(7)<>-1 THEN PRINT"Pou
r cela il faut d'abord que je l'aie.":GO
TO 1080
7649 PRINT m$(0):GOTO 1080
7650 INPUT"Quelle page ";se
7651 IF se<>co THEN PRINT m$(1):GOTO 108
0
7652 CLS:PRINT"CONSEILS POUR SITUATION D
E DETRESSE     PARTICULIERE:":PRINT STRI
NG$(40,154):PRINT
7654 PRINT"S'il devait t'arriver que les
 tiens     soient plonges dans un long s
ommeil par un sorcier mal intentionne, a
lors ecoutemon conseil:":PRINT
7657 PRINT:PRINT"Procure-toi les quatre
ingredients de    l'elixir de Humbug et a
pporte-les au     sorcier de l'autre cote
 de la foret      enchantee."
7660 PRINT:PRINT"Si tu as sa faveur, il
t'aidera."
7661 PRINT:PRINT"S'il est contre toi il
te reste a        esperer qu'un prince re
veille d'un       baiser ta fille et avec
 elle tous les    habitants du chateau."
7666 entree$=INKEY$:IF entree$="" THEN 7
666 ELSE CLS:GOTO 1080
7900 PRINT m$(0):GOTO 1080
8000 IF o=17 AND ob(16)<>-1 THEN PRINT"A
vec quoi ?":GOTO 1080
```

```
8001 IF o=17 AND ob(16)=-1 THEN PRINT ok
$:passage(6,4)=7:fl(3)=-1:GOTO 1080
8002 IF o=35 AND ob(35)=-1 THEN PRINT"Il
 y avait dedans un liquide epais.":GOTO
1080
8099 PRINT m$(0):GOTO 1080
9000 IF o=9 AND ob(14)=-1 AND joueur=5 T
HEN PRINT"Les haubans ont ete coupes.":f
l(1)=-1:GOTO 1080
9001 IF o=9 AND ob(14)<>-1 AND joueur=5
THEN PRINT"Avec quoi ?":GOTO 1080
9999 PRINT m$(0):GOTO 1080
10000 IF ob(o)<>joueur AND ob(o)<>-1 THE
N PRINT"Je ne vois rien de tel ici.":GOT
O 1080
10001 IF o=9 THEN PRINT"Pour cela il fau
t au moins deux hommes.":GOTO 1080
10002 IF o=14 AND joueur=5 THEN PRINT"Le
s haubans ont ete coupes.":fl(1)=-1:GOTO
 1080
10003 IF o=31 THEN PRINT"D'accord - je s
uis dans l'armure.":ob(31)=-1:GOTO 1080
10990 PRINT m$(0):GOTO 1080
11000 IF o=5 AND joueur=24 AND ob(5)=-1
THEN PRINT ok$:ob(o)=0:ob(41)=-1:GOTO 10
80
11001 IF o=5 AND joueur<>24 AND ob(5)=-1
 THEN PRINT"Avec quoi ?":GOTO 1080
11990 PRINT m$(0):GOTO 1080
12000 IF ob(o)<>-1 THEN GOTO 12900
12001 IF o=4 THEN PRINT ok$:ob(o)=joueur
:GOTO 1080
12002 IF o=5 THEN PRINT ok$:ob(o)=joueur
:GOTO 1080
12003 IF o=7 THEN PRINT ok$:ob(o)=joueur
:GOTO 1080
12004 IF o=12 THEN PRINT ok$:ob(o)=joueu
r:GOTO 1080
12005 IF o=15 THEN PRINT ok$:ob(o)=joueu
r:GOTO 1080
```

```
12006 IF o=16 THEN PRINT ok$:ob(o)=joueu
r:GOTO 1080
12007 IF o=14 THEN PRINT ok$:ob(o)=joueu
r:GOTO 1080
12008 IF o=30 THEN PRINT ok$:ob(o)=joueu
r:GOTO 1080
12009 IF o=41 THEN PRINT ok$:ob(o)=joueu
r:GOTO 1080
12010 IF o=45 THEN PRINT ok$:ob(o)=joueu
r:GOTO 1080
12011 IF o=40 THEN PRINT ok$:ob(o)=joueu
r:GOTO 1080
12012 IF o=36 THEN PRINT ok$:ob(o)=joueu
r:GOTO 1080
12013 IF o=35 THEN PRINT ok$:ob(o)=joueu
r:GOTO 1080
12014 IF o=31 THEN PRINT ok$:ob(o)=joueu
r:GOTO 1080
12900 IF o=5 AND ob(41)=-1 THEN PRINT ok
$:ob(41)=joueur:GOTO 1080
12901 IF o=16 AND ob(34)=-1 THEN PRINT o
k$:ob(34)=joueur:GOTO 1080
12990 PRINT m$(0):GOTO 1080
13000 IF ob(30)=-1 THEN ob(30)=20:PRINT
m$(6):RETURN
13010 IF ob(35)=-1 THEN ob(35)=20:PRINT
m$(6):RETURN
13020 IF ob(5)=-1 THEN ob(5)=20:PRINT m$
(6):RETURN
13090 RETURN
14000 IF ob(34)<>25 THEN RETURN
14010 IF ob(35)<>25 THEN RETURN
14020 IF ob(41)<>25 THEN RETURN
14030 IF ob(40)<>25 THEN RETURN
14100 CLS:PRINT"Une voix retentit:":PRIN
T:PRINT
14110 PRINT"Le grand Humbug se rejouit d
e tes dons. Il mettra sa magie a ton ser
vice.":PRINT:PRINT
```

```
14140 PRINT"Rentre maintenant a la maiso
n car une   fete s'y prepare pour feter
le reveil   apres le long sommeil."
14160 GOTO 14160
15100 IF ob(31)<>-1 THEN RETURN
15110 m$(0)="Je suis trop lourd et je co
ule.":GOTO 4500
15200 IF RND(1)>0.3 THEN RETURN
15210 m$(0)="Je suis tombe dans un piege
.":GOTO 4500
16000 IF o=10 AND joueur=8 THEN PRINT"Je
 n'y arrive pas.":GOTO 1080
16010 IF o=11 AND joueur=8 THEN PRINT"Je
 n'y arrive pas.":GOTO 1080
16020 PRINT m$(0):GOTO 1080
```

L'IVRESSE DE L'OR

Point n'est besoin de parler beaucoup pour présenter ce jeu. Il s'agit quand même ici de la version complète, l'aventure ne commencera véritablement que lorsque vous serez parvenu à quitter le territoire que l'on connaît jusqu'ici.

Vous pénétrerez dans le monde obscur d'une vieille mine d'or et vous demanderez plus d'une fois si vous voulez agir spontanément et laisser s'écouler ainsi un temps précieux, mesuré ici par la durée d'éclairage d'une lampe, ou bien si vous préférez agir à coup sûr en n'exécutant que des actions justifiées par la réflexion.

```
  1 REM ********************************
**
2 REM * L'ivresse de l'or, Ver 1      *
3 REM * (c) 1984 by Walkowiak         *
4 REM ********************************

5 DEFINT a-z:'ON ERROR GOTO 1780
10 MODE 1:PAPER 0:INK 0,0:BORDER 0:PEN 1
:INK 1,25:PEN 2:INK 2,24
20 PLOT 420, 170:DRAW 420, 360:DRAW 640,
 360:DRAW 640, 340:DRAW 450, 340:DRAW 45
0, 170:DRAW 460, 180:DRAW 460, 340:PLOT
460, 200:DRAW 530, 270:PLOT 530, 340:DRA
W 530, 260:DRAW 540, 260:DRAW 540, 340:P
LOT 540, 260
21 DRAW 550, 270:DRAW 550, 340:PLOT 550,
 290:DRAW 600, 340:PLOT 640, 340:DRAW 30
0, 0:PLOT 440, 0:DRAW 640, 200:PLOT 640,
 200:PLOT 500, 200:DRAW 640, 200:PLOT 60
0, 160:DRAW 460, 160:PLOT 420, 120:DRAW
560, 120
22 PLOT 520, 80:DRAW 380, 80:PLOT 340, 4
0:DRAW 480, 40:PLOT 440, 0:DRAW 300, 0:P
LOT 540, 240:DRAW 640, 240:PLOT 580, 280
:DRAW 640, 280:PLOT 620, 320:DRAW 640, 3
20:PLOT 450, 170:DRAW 420, 170:PLOT 420,
 360
23 DRAW 440, 380:DRAW 640, 380:PLOT 430,
 370:DRAW 430, 380:DRAW 440, 390:DRAW 44
0, 400:PLOT 280, 150:DRAW 150, 150:DRAW
150, 200:DRAW 280, 200:DRAW 280, 150:DRA
W 310, 180:PLOT 310, 180:PLOT 280, 200:D
RAW 310, 230
24 DRAW 310, 180:PLOT 310, 230:DRAW 180,
 230:DRAW 150, 200:PLOT 150, 200:DRAW 16
0, 210:DRAW 270, 210:DRAW 280, 200:PLOT
270, 210:DRAW 290, 230:PLOT 270, 190:DRA
W 160, 190:DRAW 160, 160:DRAW 270, 160:D
RAW 270, 190
25 PRINT CHR$(150);STRING$(18,154);CHR$(
156)
```

```
26 PRINT CHR$(149);STRING$(18,32);CHR$(1
49)
27 PRINT CHR$(149);"L'IVRESSE DE L'OR ";
CHR$(149)
28 PRINT CHR$(149);STRING$(18,32);CHR$(1
49)
29 PRINT CHR$(149);"un jeu d'aventure ";
CHR$(149)
30 PRINT CHR$(149);"de J";CHR$(178);"rg
Walkowiak ";CHR$(149)
31 PRINT CHR$(149);STRING$(18,32);CHR$(1
49)
32 PRINT CHR$(147);STRING$(18,154);CHR$(
153)
33 LOCATE 3,25:PRINT CHR$(164);" 1984  b
y DATA BECKER, Duesseldorf";
40 FOR i=1 TO 8000:NEXT
110 AR=31
120 ao=45
130 av=11
140 af=7
150 joueur=1
160 longueurmot=4
170 wmax=60
171 evaluation=0
180 coup=0
185 imax=4
186 lm=0:lumiere=-1:lw=1:l1=20
190 DIM lieu$(ar),passage(ar,6),ob$(ao),
nom$(ao),ob(ao),verb$(av)
200 REM ************************ VERBES
201 DATA examine
202 DATA prends
203 DATA pose
204 DATA ouvre
205 DATA utilise
206 DATA detruis
207 DATA allume
208 DATA remplis
```

```
209 DATA entre
210 DATA supprime
211 DATA fixe
300 REM ******************** OBJETS
301 DATA"beaucoup de grands arbres","arb
res",1
302 DATA"beaucoup de grands arbres","arb
res",2
303 DATA"quelques rochers","rochers",2
304 DATA"une cabane en bois en ruine","c
abane",3
305 DATA"une bonbonne sale","bonbonne",0
306 DATA"du miel","miel",0
307 DATA"une caisse en bois","caisse",3
308 DATA"une etagere branlante","etagere
",0
309 DATA"un peu d'explosif","explosif",0
310 DATA"un sombre puits naturel","puits
",0
311 DATA"un coffre de fer rouille","coff
re",0
312 DATA"*des pieces d'argent*","pieces"
,0
313 DATA"une sombre grotte dans la roche
","grotte",5
314 DATA"un ours a l'air mechant","ours"
,0
315 DATA"d'innombrables petits arbustes"
,"arbustes",4
316 DATA"de nombreuses barres de fer","b
arres",6
317 DATA"*des pepites*","pepite",5
318 DATA"une barre de fer","fer ",0
319 DATA"une chaine de fer","chaine",0
320 DATA des rails rouilles,rails,7
321 DATA une corde,corde,0
322 DATA un lourd piolet,piolet,8
323 DATA une vieille lampe,lampe,8
```

```
324 DATA le bout de la galerie,bout,9
325 DATA un crochet de fer,crochet,0
326 DATA une mine,mine,9
327 DATA une cloison de planches,cloison
,10
328 DATA des decombres,decombres,11
329 DATA de vieux sacs,sacs,0
330 DATA des parois de pierre humides,pa
rois,13
331 DATA un liquide nauseabond,liquide,1
7
332 DATA un cadavre,cadavre,18
333 DATA *des blocs d'argent*,blocs,0
334 DATA *des louis d'or*,louis,0
335 DATA des toiles d'araignee,toiles,20
336 DATA un mur d'or,mur,22
337 DATA une pancarte,pancarte,22
338 DATA *de l'or*,"or  ",31
339 DATA une rive de lac,rive,30
340 DATA le lac,"lac ",30
341 DATA des minerais d'or,minerai,0
342 DATA une meche,meche,0
343 DATA un luntero,luntero,-1
344 DATA -,paradis,0
345 DATA un vieux wagonnet,wagonnet,7
500 REM ********** DESCRIPTIONS DE LIEUX
501 DATA"dans la foret",1,1,1,2,0,0
502 DATA"dans la foret",2,1,1,3,0,0
503 DATA"dans la foret devant un versant
 de rocher",0,4,2,0,0,0
504 DATA"dans une clairiere",3,0,5,0,0,0
505 DATA"dans une clairiere",0,0,6,4,0,0
506 DATA"devant une galerie de mine.",7,
0,1,5,0,0
507 DATA dans la galerie d'entree,8,6,0,
0,0,0
508 DATA dans un renfoncement de la gale
rie,9,7,8,10,0,0
509 DATA au bout de la galerie,0,8,0,0,0
,0
```

```
510 DATA dans une galerie secondaire,11,
0,8,0,0,0
511 DATA dans une vieille mine,0,10,0,0,
0,0
512 DATA dans la grotte,0,5,0,13,0,0
513 DATA dans la grotte,0,0,12,14,0,0
514 DATA dans la grotte,0,16,13,15,0,0
515 DATA dans la grotte,0,0,14,0,0,0
516 DATA dans une galerie,14,0,0,17,0,0
517 DATA dans une grotte secondaire,0,0,
16,0,0,0
518 DATA sur le sol de la mine,0,0,19,0,
0,0
519 DATA dans une galerie sinueuse,24,0,
20,18,0,0
520 DATA dans une galerie large,0,0,21,1
9,0,0
521 DATA dans une vielle voie de taille,
22,11,11,20,11,0
522 DATA dans une coupole rocheuse,0,21,
0,0,27,0
523 DATA dans un paradis souterrain,0,0,
0,0,0,0
524 DATA dans une galerie sinueuse,26,19
,24,24,24,24
525 DATA dans une galerie sinueuse,27,24
,24,29,26,24
526 DATA dans une galerie sinueuse,27,24
,26,27,27,24
527 DATA dans une galerie sinueuse,25,24
,26,27,26,24
528 DATA dans une galerie sinueuse,24,26
,26,27,0,0
529 DATA dans une galerie sinueuse,0,27,
30,27,0,0
530 DATA vor einem unterirdischen See,28
,0,0,0,0,0
531 DATA dans une petite grotte,30,0,0,0
,0,0
```

```
600 REM ******************** MESSAGES
601 m$(1)="Je ne vois rien de special
602 m$(2)="Je ne suis pas si fort !"
603 m$(3)="Comment vois-tu cela ?"
604 m$(4)="L'ours prend le miel et dispa
rait"
605 m$(5)="dans les profondeurs de la gr
otte."
690 ok$="D'accord"
699 REM *********** 2eme TITRE: INTRODUC
TION
700 CLS:PEN 1:PRINT"   B I E N V E N U E
   D A N S   L A":PRINT:PEN 2:PRINT TAB(
15)CHR$(34);"RUEE VERS L'OR";CHR$(34):PR
INT:PEN 1
705 PRINT STRING$(40,208)
710 PRINT"A la poursuite de la fortune d
ans le    nouveau monde, vous avez renco
ntre il y a quelques jours un vieillard
agonisant auquel vous avez porte assista
nce dans  ses derniers moments."
720 PRINT"Par reconnaissance il vous a p
arle de savieille mine d'or et des reste
s de son  tresor qui y sont caches."
730 PRINT"Sur le chemin de cette mine vo
us avez   echappe a d'innombrables peril
s. Vous   aurez bientot atteint votre bu
t et vous saurez enfin si le vieillard a
 dit vrai ou s'il delirait."
740 PRINT STRING$(40,210)
810 FOR i=1 TO av
815 READ verb$(i):verb$(i)=LEFT$(verb$(i
),longueurmot):verb$(i)=UPPER$(verb$(i))
820 NEXT i
830 FOR objet=1 TO ao
835 READ ob$(objet),nom$(objet),ob(objet
):nom$(objet)=LEFT$(nom$(objet),longueur
mot):nom$(objet)=UPPER$(nom$(objet))
```

```
840 NEXT objet
845 FOR lieu=1 TO AR
850 READ lieu$(lieu)
855 FOR direction=1 TO 6
860 READ passage(lieu,direction)
865 NEXT direction
870 NEXT lieu
880 PRINT:INPUT"  Voulez-vous des consei
ls";entree$:entree$=UPPER$(entree$)
890 IF LEFT$(entree$,1)="O" THEN GOSUB 9
00
895 GOTO 1000
899 REM ******** 3eme TITRE: INSTRUCTION
S
900 MODE 1:PAPER 0:INK 0,0:BORDER 0:PEN
1:INK 1,25
910 PRINT"       CPC - Ventures"
920 PRINT TAB(15)CHR$(164);" 1984  by J"
;CHR$(178);"rg Walkowiak"
925 PRINT STRING$(40,208):PRINT"Imaginez
 un robot que vous pourriez     commande
r avec de nombreux ordres."
930 PRINT"Je suis ce robot et je vais m'
exposer   pour vous aux aventures les pl
us        risquees."
940 PRINT"Pour que vous puissiez me diri
ger       utilement je vous decrirai cha
que fois  precisement la situation dans
laquelle  je me trouve."
950 PRINT"Vous me direz ensuite avec deu
x mots    comme par exemple PRENDS COUTE
AU ce que je dois faire.":PRINT
960 PRINT"Je comprends en outre les inst
ructions  INVENTAIRE, SCORE, VOCABULAIRE
, HELP    SAVE & LOAD ainsi que END."
970 PRINT STRING$(40,210)
980 INK 3,12,24:INK 2,24,12:PEN 3: PRINT
"        Appuyez";:PEN 2:PRINT" <sur une
 TOUCHE>";:PEN 1:PRINT" ..."
```

```
990 entree$=INKEY$:IF entree$="" THEN 99
0 ELSE CLS:MODE 1:INK 1,2:INK 2,14:INK 3
,26:RETURN
1000 MODE 1:PAPER 0:INK 0,0:BORDER 0:PEN
 1:INK 1,2:PEN 2:INK 2,14:PEN 3:INK 3,26

1010 lignevide$=STRING$(40," ")
1020 DATA le Nord, le Sud, l'Ouest, l'Es
t, en haut, en bas
1030 FOR direction=1 TO 6
1040 READ direction$(direction)
1050 NEXT direction
1070 CLS
1080 PRINT:PRINT:coup=coup+1:lw=lw+1
1084 IF lw=lm THEN GOSUB 3000
1085 IF evaluation=wmax THEN GOTO 4800
1090 LOCATE 1,1
1091 REM ******************** PAS DE LUM
IERE
1092 IF lumiere=-1 THEN 1100
1093 PRINT"Je ne sais pas exactement ou
je suis.   Il fait trop sombre pour y vo
ir quoi quece soit."
1095 PRINT:PRINT"Je ne vois meme plus le
s issues !":GOTO 1330
1100 FOR ligne=1 TO 10
1110 PRINT lignevide$;
1120 NEXT ligne
1130 LOCATE 1,1:PEN 1
1140 PRINT"Je suis ";
1150 PRINT lieu$(joueur);:PRINT"."
1160 PRINT"Je vois ";
1170 FOR i=1 TO ao
1180 IF ob(i)<>joueur THEN 1210
1190 IF POS(#0)+LEN(ob$(i))+2<=39 THEN P
RINT ob$(i);", ";:GOTO 1210
1200 IF POS(#0)+LEN(ob$(i))+2>39 THEN PR
INT:GOTO 1190
1210 NEXT i
```

```
1220 PRINT CHR$(8);CHR$(8);"."
1230 PRINT lignevide$:PEN 2
1240 PRINT"Je peux aller vers ";:imprime
=0
1250 FOR direction=1 TO 6
1260 IF passage(joueur,direction)=0 THEN
 GOTO 1300 ELSE imprime=-1
1270 IF POS(#0)=20 THEN PRINT direction$
(direction);:GOTO 1300
1280 IF POS(#0)+LEN(direction$(direction
))<38 THEN PRINT", ";direction$(directio
n);:GOTO 1300
1290 PRINT",":PRINT direction$(direction
);:GOTO 1300
1300 NEXT direction
1310 IF imprime=0 THEN PRINT"nulle part"
;
1320 PRINT".":PEN 3
1330 PRINT"-----------------------------
----------"
1340 IF evaluation=wmax THEN GOTO 4800
1350 IF joueur=15 THEN m$(0)="Malheureus
ement c'est la que l'ours     etait.":FO
R i=1 TO 1000:NEXT i:GOTO 4500
1360 IF lw<=0 THEN lumiere=0
1370 IF ex=coup AND joueur=20 THEN m$(0)
="Aie ! Moi aussi j'ai ete dechiquete.":
GOTO 4500
1380 IF joueur>18 AND ob(23)<>-1 THEN lu
miere=0
1390 PEN 3:LOCATE 1,25:INPUT"Que dois-je
 faire";entree$:entree$=UPPER$(entree$):
PEN 2
1395 rl=lm-lw:IF (rl>0 AND rl<15) THEN P
RINT"dans";lm-lw;"coups je serai dans l'
obscurite !"
1400 IF LEN(entree$)>2 THEN 1500
1410 IF entree$="N" AND passage(joueur,1
)<>0 THEN joueur=passage(joueur,1):PRINT
"D'accord":GOTO 1080
```

```
1420 IF entree$="S" AND passage(joueur,2
)<>0 THEN joueur=passage(joueur,2):PRINT
"D'accord":GOTO 1080
1430 IF entree$="O" AND passage(joueur,3
)<>0 THEN joueur=passage(joueur,3):PRINT
"D'accord":GOTO 1080
1440 IF entree$="E" AND passage(joueur,4
)<>0 THEN joueur=passage(joueur,4):PRINT
"D'accord":GOTO 1080
1450 IF entree$="H" AND passage(joueur,5
)<>0 THEN joueur=passage(joueur,5):PRINT
"D'accord":GOTO 1080
1460 IF entree$="B" AND passage(joueur,6
)<>0 THEN joueur=passage(joueur,6):PRINT
"D'accord":GOTO 1080
1470 IF LEN(entree$)<3 THEN PRINT"Aucun
chemin n'y mene !":GOTO 1080
1480 IF LEN(entree$)>8 THEN GOTO 2000
1499 REM ********** DEBUT INVENTAI
RE
1500 IF LEFT$(entree$,3)<>"INV" THEN GOT
O 1560
1510 PRINT"Voici ce que je porte sur moi
:"
1520 FOR objet=1 TO ao
1530 IF ob(objet)=-1 THEN PRINT ob$(obje
t)
1540 NEXT objet
1550 GOTO 1080
1560 IF LEFT$(entree$,3)<>"SCO" THEN GOT
O 1600
1561 PRINT"Sur";wmax;"points, tu as obte
nu en":PRINT coup;"coups:":PRINT evaluat
ion;"points !":PRINT"Cela correspond a u
ne moyenne de";evaluation/coup;"points."
:GOTO 1080
1599 REM ********** SAVE GAME
1600 IF LEFT$(entree$,3)<>"SAV" THEN GOT
O 1700
```

```
1610 PRINT"Appuyer sur PLAY & REC ...":P
RINT"Sauvegarder sous quel nom ...";STRI
NG$(4,8);:INPUT entree$
1620 IF LEN(entree$)>10 THEN PRINT"Un pe
u plus court S.V.P !":GOTO 1610 ELSE ent
ree$="!"+entree$+".JEU":OPENOUT entree$
1625 PRINT#9,joueur
1630 FOR objet=1 TO ao
1631 PRINT#9,ob(objet)
1632 NEXT objet
1635 FOR lieu=1 TO ar
1636 FOR direction=1 TO 6
1637 PRINT#9,passage(lieu,direction)
1638 NEXT direction
1639 NEXT lieu
1645 FOR flag=1 TO af
1646 PRINT#9,fl(flag)
1647 NEXT flag
1650 CLOSEOUT
1660 PRINT ok$
1670 GOTO 1080
1699 REM ********************* LOAD GAME
1700 IF LEFT$(entree$,3)<>"LOA" THEN GOT
O 1800
1710 PRINT"Rembobiner cassette et appuye
r sur PLAY":PRINT"Quel jeu faut-il charg
er ...";STRING$(4,8);:INPUT entree$
1720 IF LEN(entree$)>10 THEN PRINT"Ce n'
est pas possible !":GOTO 1710 ELSE entre
e$="!"+entree$+".JEU":OPENIN entree$
1725 INPUT#9,joueur
1730 FOR objet=1 TO ao
1731 INPUT#9,ob(objet)
1732 NEXT objet
1735 FOR lieu=1 TO ar
1736 FOR direction=1 TO 6
1737 INPUT#9,passage(lieu,direction)
1738 NEXT direction
1739 NEXT lieu
1745 FOR flag=1 TO af
```

```
1746 INPUT#9,fl(flag)
1747 NEXT flag
1750 CLOSEIN
1760 PRINT ok$
1770 GOTO 1080
1780 PRINT"Attention erreur !":BORDER 3:
RESUME NEXT
1800 IF LEFT$(entree$,3)<>"VOC" THEN GOT
O 1900
1805 CLS:PRINT"Je comprends les verbes s
uivants:":PRINT:PRINT:RESTORE
1810 FOR i=1 TO av
1820 READ mot$:PRINT mot$
1830 NEXT i
1840 GOSUB 1890
1845 CLS:PRINT"et je connais les objets
suivants:":PRINT
1849 ligne=0
1850 FOR i=1 TO ao
1855 ligne=ligne+1
1860 READ mot$,mot$,x:PRINT mot$
1865 IF ligne=20 THEN GOSUB 1890
1866 IF ligne=20 THEN ligne=1:CLS
1870 NEXT i
1880 GOSUB 1890:CLS:GOTO 1080
1890 LOCATE 1,25:PRINT "Appuyer sur une
touche"
1895 entree$=INKEY$:IF entree$="" THEN 1
895 ELSE CLS:RETURN
1900 IF LEFT$(entree$,3)<>"INS" THEN GOT
O 1950
1910 GOSUB 900
1920 GOTO 1080
1950 IF LEFT$(entree$,3)<>"END" THEN GOT
O 1970 ELSE CLS
1960 PRINT"Nous esperons que vous aurez
plus de    succes la prochaine fois !":P
RINT:PRINT:PRINT:END
1970 IF LEFT$(entree$,3)<>"HEL" THEN GOT
O 2000
```

```
1971 IF joueur=4 AND ob(10)=0 THEN PRINT
"J'ai failli tomber dans une fosse":GOTO
 1080
1972 IF joueur=4 AND ob(11)<>joueur AND
NOT fl(2) THEN PRINT"Il me faut quelque
chose pour briser la chaine !":GOTO 1080
1975 PRINT"D'abord voir, puis reflechir
!":PRINT"Et enfin agir.":GOTO 1080
1979 REM ******************** FIN HELP
2000 longueur=LEN(entree$)
2010 FOR lettre=1 TO longueur
2020 tester$=MID$(entree$,lettre,1)
2030 IF tester$<>" "THEN NEXT lettre
2040 everb$=LEFT$(entree$,longueurmot)
2050 rl=longueur-lettre
2060 IF rl<0 THEN 2090
2070 eobjet$=RIGHT$(entree$,rl)
2080 eobjet$=LEFT$(eobjet$,longueurmot)
2090 FOR numeroverb=1 TO av
2100 IF everb$=verb$(numeroverb) THEN 21
30
2110 NEXT numeroverb
2120 PRINT"Je ne comprends pas ce verbe
!":GOTO 1080
2130 FOR o=1 TO ao
2140 IF eobjet$=nom$(o) THEN 2200
2150 NEXT o
2160 PRINT"Je ne connais pas cet objet !
":GOTO 1080
2190 REM                  exam pren pos
ouvre utilise detruis
2200 ON numeroverb GOTO 5000,2210,7000,8
000,9000,10000,11000,12000,13000,14000,1
5000
2209 REM ***************** LIMITER INV
2210 nombre=0:FOR i=1 TO ao
2220 IF ob(i)=-1 THEN nombre=nombre+1
2230 IF nombre=imax THEN PRINT"Non merci
! Je porte deja assez de       choses.":
GOTO 1080
```

```
2240 NEXT i
2250 GOTO 6000
2260 REM ****************** fin test inv
2999 REM UP commutateur
3000 IF lumiere=-1 THEN lumiere=0:GOTO 3
020
3010 IF lumiere=0 THEN lumiere=-1
3020 lw=0:RETURN
4500 CLS:REM                    JOUEUR MORT
4600 PRINT"Meme cela !":PRINT:PRINT m$(0
)
4610 PRINT:PRINT"Je suis mort !":PRINT
4620 INPUT"Dois-je essayer encore une fo
is ";entree$:entree$=UPPER$(entree$)
4630 IF LEFT$(entree$,1)="O" THEN RUN
4640 GOTO 1960
4800 CLS:REM                  JOUEUR GAGNE
4810 PRINT"Toutes nos felicitations !"
4820 PRINT:PRINT"Vous avez resolu l'epre
uve qui vous     etait imposee et vous p
ouvez maintenant vous lancer dans une au
tre aventure."
4830 PRINT:PRINT:PRINT:PRINT:PRINT:END
4999 REM *********** JOUER UN COUP
5000 IF ob(o)<>joueur AND ob(o)<>-1 THEN
 GOTO 5900
5002 IF o=1 THEN PRINT m$(1):GOTO 1080
5003 IF o=3 THEN PRINT m$(1):GOTO 1080
5004 IF o=4 THEN PRINT"Il y a une etager
e dans un coin.":ob(8)=joueur:GOTO 1080
5005 IF o=5 THEN PRINT"La bonbonne est p
leine de miel.":GOTO 1080
5006 IF o=6 THEN PRINT m$(1):GOTO 1080
5007 IF o=7 AND ob(9)=0 THEN PRINT"Il y
a des explosifs dans la caisse.":GOTO 10
80
5008 IF o=8 AND ob(5)=0 THEN PRINT"Il y
a une bonbonne sur l'etagere.":ob(5)=jou
eur:GOTO 1080
```

```
5009 IF o=8 AND ob(5)<>0 THEN PRINT m$(1
):GOTO 1080
5010 IF o=10 THEN PRINT"Il y a un coffre
 d'acier dans le puits.":ob(11)=joueur:G
OTO 1080
5011 IF o=11 AND NOT fl(1) THEN PRINT"Il
 est ferme par une chaine de fer.":ob(19
)=joueur:GOTO 1080
5012 IF o=11 AND fl(1) AND NOT fl(2) THE
N PRINT"Je ne vois rien de special
        exterieurement.":GOTO 1080
5013 IF o=11 AND fl(1) AND fl(2) AND ob(
12)=0 THEN PRINT"Il est plein de pieces
d'argent.":ob(12)=joueur:GOTO 1080
5014 IF o=12 THEN PRINT"C'est exactement
 ce que je cherche !":GOTO 1080
5015 IF o=13 THEN PRINT"J'ai effraye un
ours !":ob(14)=joueur:fl(3)=0:GOTO 1080
5017 IF o=15 THEN PRINT"Il y a un puits
naturel entre les       arbustes.":ob(10
)=joueur:GOTO 1080
5018 IF o=16 THEN PRINT"Elles ont l'air
tres stables.":GOTO 1080
5019 IF o=17 THEN PRINT"C'est de l'or pu
r !":GOTO 1080
5020 IF o=45 AND ob(21)=0 THEN PRINT"Le
cable y est encore accroche.":ob(21)=jou
eur:GOTO 1080
5021 IF o=23 THEN PRINT"C'est une vieill
e lampe a huile.":GOTO 1080
5022 IF o=24 THEN PRINT"Il y a un croche
t au mur.":ob(25)=joueur:GOTO 1080
5023 IF o=25 THEN PRINT"Je regrette, il
est trop enfonce.":GOTO 1080
5024 IF o=26 THEN m$(0)="C'est tres prof
ond et j'etais trop pres du bord.":GOTO
4500
5025 IF o=27 THEN PRINT"Le menuisier eta
it un expert.":GOTO 1080
```

```
5026 IF o=28 AND ob(o)=0 THEN PRINT"J'ai
 decouvert deux sacs.":ob(29)=joueur:GOT
O 1080
5027 IF o=29 THEN PRINT"Ils sont pleins
de poussiere d'or !":GOTO 1080
5028 IF o=31 THEN PRINT"Il est tellement
 huileux qu'il colle auxdoigts.":GOTO 10
80
5029 IF o=32 THEN PRINT"Il pue !"
5030 IF o=32 AND ob(34)=0 THEN PRINT"Il
y a des pieces d'or dans la poche de la
veste.":ob(34)=joueur:GOTO 1080
5031 IF o=35 THEN PRINT"Il n'y a pas de
toiles d'araignees.":ob(o)=0:ob(42)=joue
ur:PRINT"C'est une meche.":GOTO 1080
5033 IF o=36 THEN PRINT"C'est vraiment d
e l'or pur.":GOTO 1080
5034 IF o=37 THEN PRINT"Elle porte l'ins
cription:":PRINT"Lorsque l'avidite se re
veille la mort   fait alors souvent de b
onnes affaires.":GOTO 1080
5036 IF o=42 THEN PRINT"Elle va jusqu'au
 plafond.":GOTO 1080
5037 IF o=43 THEN PRINT"C'est un arme a
feu courante au Farwest.":GOTO 1080
5038 IF o=32 AND fl(4)=-1 AND ob(33)<>-2
 THEN PRINT"Il reposait sur des morceaux
 d'argent.":ob(33)=joueur:GOTO 1080
5800 PRINT m$(1):GOTO 1080
5900 REM      objet manquant
5901 IF o=1 AND joueur=2 THEN PRINT m$(1
):GOTO 1080
5902 IF o=6 AND ob(5)=-1 THEN PRINT"Il e
st doux et delicieux.":GOTO 1080
5903 IF o=6 AND ob(5)<>-1 THEN PRINT"Je
n'ai pas de miel !":GOTO 1080
5904 IF o=9 AND (ob(7)=joueur OR ob(7)=-
1) THEN PRINT"Il a l'air tres puissant !
":GOTO 1080
5905 IF o=16 AND ob(18)=-1 THEN PRINT"A
peine rouillees - vraiment stables !":GO
TO 1080
```

```
5910 IF o=14 AND joueur=5 AND ob(14)=5 T
HEN PRINT"Il semble que j'eveille son ap
petit.":GOTO 1080
5911 IF o=20 AND joueur=22 THEN PRINT"Lo
rsque l'avidite se reveille la mort  fa
it alors souvent de bonnes affaires.":GO
TO 1080
5912 IF o=44 AND joueur=23 THEN m$(0)="J
e suis au royaume des morts.":GOTO 4500
5990 PRINT"Je ne vois rien de tel ici!":
GOTO 1080
6000 IF ob(o)<>joueur AND ob(o)<>-1 THEN
 GOTO 6900
6001 IF o=1 THEN PRINT m$(2):GOTO 1080
6002 IF o=3 THEN PRINT m$(2):GOTO 1080
6003 IF o=4 THEN PRINT m$(3):GOTO 1080
6004 IF o=8 THEN PRINT m$(2):GOTO 1080
6005 IF o=11 THEN PRINT m$(2):GOTO 1080
6006 IF o=10 THEN PRINT m$(3):GOTO 1080
6007 IF o=13 THEN PRINT m$(3):GOTO 1080
6008 IF o=15 THEN PRINT m$(2):GOTO 1080
6010 IF o=5 THEN ob(5)=-1:PRINT ok$:GOTO
 1080
6011 IF o=6 THEN ob(5)=-1:PRINT ok$:GOTO
 1080
6012 IF o=7 THEN ob(o)=-1:PRINT ok$:GOTO
 1080
6014 IF o=12 THEN ob(o)=-2:evaluation=ev
aluation+10:PRINT ok$:GOTO 1080
6015 IF o=14 AND joueur=5 THEN m$(0)="L'
ours m'a tue.":GOTO 4500
6016 IF o=16 THEN ob(18)=-1:PRINT ok$:GO
TO 1080
6017 IF o=17 AND fl(3) THEN PRINT ok$:ob
(o)=-1:GOTO 1080
6018 IF o=17 AND NOT fl(3) THEN m$(0)="U
n ours se jette sur moi. Il m'a tue.":GO
TO 4500
6019 IF o=12 AND ob(o)=-2 THEN PRINT"J'a
i deja l'argent.":GOTO 1080
```

```
6020 IF o=17 AND ob(o)=-2 THEN PRINT"J'a
i deja l'or.":GOTO 1080
6021 IF o=45 THEN PRINT m$(2):GOTO 1080
6022 IF o=21 AND fl(4)=-1 THEN PRINT ok$
:ob(o)=-1:GOTO 1080
6023 IF o=22 THEN PRINT ok$:ob(o)=-1:GOT
O 1080
6024 IF o=24 THEN PRINT ok$:ob(o)=-1:GOT
O 1080
6025 IF o=25 THEN PRINT"Il est enfonce t
rop profondement dans lerocher.":GOTO 10
80
6026 IF o=27 THEN PRINT m$(3):GOTO 1080
6027 IF o=28 THEN PRINT"Et que dois-je e
n faire ?":GOTO 1080
6030 IF o=31 AND ob(5)<>-1 THEN PRINT"Av
ec quoi donc ?":GOTO 1080
6031 IF o=31 AND ob(5)=-1 THEN PRINT ok$
;" La bouteille est pleine.":fl(6)=-1:GO
TO 1080
6032 IF o=32 THEN PRINT ok$;" - oh qu'es
t-ce donc ?":fl(5)=-1:GOTO 1080
6033 IF o=33 AND ob(o)=joueur THEN PRINT
 ok$:ob(o)=-2:evaluation=evaluation+10:G
OTO 1080
6034 IF o=34 AND ob(o)=joueur THEN PRINT
 ok$:ob(o)=-2:evaluation=evaluation+10:G
OTO 1080
6035 IF o=35 THEN PRINT"Il n'y a pas de
toiles d'araignees,     c'etait une mech
e.":GOTO 1080
6037 IF o=37 THEN PRINT ok$:ob(o)=-1:GOT
O 1080
6038 IF o=41 THEN PRINT ok$:ob(o)=-1:GOT
O 1080
6039 IF o=42 THEN PRINT ok$:ob(o)=-1:GOT
O 1080
6040 IF o=43 THEN PRINT ok$:ob(o)=-1:GOT
O 1080
```

```
6041 IF o=21 AND NOT fl(4) THEN PRINT"El
le est attachee au wagonnet.":GOTO 1080
6042 IF o=23 THEN PRINT ok$:ob(o)=-1:GOT
O 1080
6043 IF o=29 AND ob(o)=joueur THEN PRINT
 ok$:ob(o)=-2:evaluation=evaluation+10:G
OTO 1080
6044 IF o=38 AND ob(o)=joueur THEN PRINT
 ok$:ob(o)=-2:evaluation=evaluation+10:G
OTO 1080
6900 IF o=9 AND (ob(7)=joueur OR ob(7)=-
1) THEN m$(0)="Les explosifs ont explose
 lorsque je lesai touches !":GOTO 4500
6901 IF o=16 AND ob(18)=joueur THEN PRIN
T ok$:ob(18)=-1:GOTO 1080
6910 IF o=1 AND joueur=2 THEN PRINT m$(2
):GOTO 1080
6911 IF o=20 AND joueur=22 THEN PRINT"Lo
rsque l'avidite se reveille la mort   fa
it alors souvent de bonnes affaires.":GO
TO 1080
6999 PRINT"Je ne vois rien de tel ici !"
:GOTO 1080
7000 IF o=16 AND ob(18)=-1 THEN PRINT ok
$:ob(18)=joueur:GOTO 1080
7001 IF ob(o)<>-1 THEN PRINT"Mais je ne
possede pas encore une telle chose !":GO
TO 1080
7010 IF o=6 AND joueur=5 THEN ob(6)=0:fl
(3)=-1:PRINT m$(4):PRINT m$(5):ob(14)=0:
GOTO 1080
7020 IF o=5 AND joueur=5 THEN ob(5)=0:fl
(3)=-1:PRINT m$(4):PRINT m$(5):ob(14)=0:
GOTO 1080
7900 ob(o)=joueur:PRINT ok$:GOTO 1080
8000 IF ob(o)<>joueur AND ob(o)<>-1 THEN
 PRINT"Il n'y a rien de tel ici !":GOTO
1080
```

```
8005 IF o=4 AND joueur=3 THEN PRINT"La c
abane etait deja ouverte.":GOTO 1080
8010 IF o=5 THEN PRINT ok$:GOTO 1080
8020 IF o=11 AND NOT fl(1) THEN PRINT"La
 chaine ne le permet pas.":GOTO 1080
8025 IF o=11 AND fl(1) THEN PRINT ok$;"
le couvercle s'ouvre et retombe en
arriere.":fl(2)=-1:GOTO 1080
8030 IF o=23 THEN PRINT ok$:GOTO 1080
8035 IF o=29 THEN PRINT ok$:GOTO 1080
8040 IF o=32 THEN PRINT"Je regrette - Je
 ne suis pas Frankenstein !":GOTO 1080
8045 IF o=36 THEN PRINT"Comment ?":GOTO
1080
8999 PRINT"Je ne comprends pas ce que tu
 veux dire.":GOTO 1080
9000 IF o=16 AND ob(18)=-1 AND joueur=4
THEN PRINT"La chaine se brise.":fl(1)=-1
:GOTO 1080
9005 IF ob(o)<>joueur AND ob(o)<>-1 THEN
 GOTO 9900
9010 IF o=16 AND joueur=4 THEN PRINT"La
chaine se brise.":fl(1)=-1:GOTO 1080
9020 IF o=45 THEN PRINT"Comment et dans
quel but ?":GOTO 1080
9030 IF o=21 THEN PRINT"Comment et dans
quel but ?":GOTO 1080
9999 PRINT"Je ne comprends pas ce que tu
 veux dire.":GOTO 1080
10000 IF o=16 AND joueur=4 AND ob(18)=-1
 THEN PRINT"La chaine se brise.":fl(1)=-
1:GOTO 1080
10010 IF o=16 AND joueur=4 AND ob(18)<>-
1 THEN PRINT"Avec quoi ?":GOTO 1080
10020 IF o=36 AND ob(22)=-1 THEN PRINT o
k$:passage(22,1)=23:GOTO 1080
10030 IF o=21 AND (ob(o)=joueur OR ob(o)
=-1) THEN PRINT ok$:fl(4)=-1:GOTO 1080
```

```
10040 IF o=27 AND ob(22)=-1 THEN PRINT o
k$:passage(10,2)=12:GOTO 1080
10050 IF o=27 AND ob(22)<>-1 THEN PRINT"
Avec quoi ?":GOTO 1080
10999 PRINT"Je ne comprends pas ce que t
u veux dire.":GOTO 1080
11000 IF ob(43)<>-1 THEN PRINT"Je n'ai r
ien pour allumer.":GOTO 1080
11010 IF o=8 AND lz<=0 THEN PRINT"Il n'y
 a plus d'huile dans la lampe !":GOTO 10
80
11020 IF o=42 AND (ob(43)=-1 OR lumiere=
-1) THEN PRINT"psssss... !":fl(7)=-1:ex=
coup+3:ob(o)=0:passage(30,2)=31:passage(
31,1)=30:GOTO 1080
11030 IF o=23 AND ob(23)=-1 THEN PRINT o
k$;" - la lampe brule.":lm=l1:lw=1:GOTO
1080
11040 IF o=43 THEN PRINT ok$;" - la mech
e luit.":GOTO 1080
11098 PRINT"Je ne te comprends pas.":GOT
O 1080
12000 IF o=23 AND (fl(6) AND ob(23)=-1)
THEN lm=60:fl(6)=0:lw=0:PRINT ok$:GOTO 1
080
12020 IF o=5 AND joueur=17 THEN fl(6)=-1
:PRINT"La bouteille est pleine.":GOTO 10
80
```

```
12030 IF o=43 AND ob(o)=-1 THEN PRINT"la
 meche luit.":GOTO 1080
12998 PRINT"Je ne comprends pas ce que t
u veux dire.":GOTO 1080
13000 IF o=13 AND joueur=5 THEN joueur=1
2:PRINT ok$:GOTO 1080
13998 PRINT"Je ne sais pas ce que tu veu
x.":GOTO 1080
14000 IF o=23 AND ob(o)=-1 THEN l1=lm-lw
:PRINT ok$:GOTO 1080
14998 PRINT"Je ne sais pas ce que tu veu
x.":GOTO 1080
15000 IF o=21 AND joueur=9 THEN PRINT ok
$:ob(o)=9:passage(9,6)=18:passage(18,5)=
9:GOTO 1080
15998 PRINT"Je ne sais pas ce que tu veu
x.":GOTO 1080
```

SPACE MISSION

Je vous en prie, ne vous laissez pas entraîner par l'attrait d'un jeu d'aventure graphique à entrer ce programme si que vous êtes totalement débutant dans ce domaine et si vos connaissances en programmation ne sont pas non plus assez étendues pour que vous puissiez chercher dans le listing les réponses aux questions qui se poseront rapidement à vous.

Car au contraire des jeux d'aventure habituels où des situations dangereuses ne naissent qu'à la suite de vos actions, vous ne disposez dans Space Mission, après entrée de RUN, que d'un temps égal à neuf (9) coups de jeu pour sauver votre vie.

Et il serait vraiment dommage de perdre de cette façon la joie de jouer pour la seule raison que vous n'avez pas gagné !

```
1 REM space mission
2 REM (c) 1984 by Walkowiak
3 REM ********************************
10 MODE 1:LOCATE 14,22:PRINT"SPACE MISSI
ON"
20 LOCATE 10,23:PRINT"(c) 1984 by Walkow
iak"
30 FOR i=1 TO 8000:NEXT
150  ar= 30:ao= 50:av= 10:af= 11:imax=2:
coup=0
160  joueur= 11
170  wl= 4
190  DIM lieu$(ar),passage(ar,6),ob$(ao)
,nom$(ao),ob(ao),verb$(av),fl(af)
200  REM ************************ VERBE
S
210  DATA examine
211  DATA prends
212  DATA ouvre
213  DATA appuie
214 DATA pose
215  DATA porte
216  DATA remplis
217  DATA fixe
218  DATA utilise
219  DATA echange
300  REM ******************** OBJETS
301  DATA le pupitre de commande,pupitre
,1
302  DATA un ecran video,ecran,1
303  DATA ,commutateur,0
304  DATA un terminal,terminal,12
305  DATA un hypercomm,hypercomm,0
306  DATA un telecomm,telecomm,0
307 DATA ,antigravite,0
308  DATA une entree,entree,0
309  DATA une entree,entree,0
310  DATA une gigantesque cage de tole,c
age,7
```

```
311  DATA l'ordinateur central,ordinateu
r,0
312  DATA une cartouche enfichable,carto
uche,-1
313  DATA des modules,modules,0
314  DATA une clef,clef,0
315  DATA des cables,cable,7
316  DATA des tuyaux,tuyaux,21
317  DATA un panneau indicateur,panneau,
8
318  DATA ,LAND,0
319  DATA une armoire murale,armoire,10
320  DATA ,carte,0
321  DATA une porte,porte,9
322  DATA le pilotage automatique,pilota
ge,0
323  DATA un bouton de commutation,bouto
n,9
324  DATA une couchette,couchette,10
325  DATA rien de special,table,11
326  DATA ,MOVE,0
327  DATA plusieurs reservoirs,reservoir
s,21
328  DATA une bouteille d'oxygene,oxygen
e,20
329  DATA un scaphandre,scaphandre,0
330  DATA une clefde10,clefde10,0
331  DATA ,STOP,0
332  DATA un tournevis,tournevis,0
333  DATA le poste de commande,poste,14
334  DATA une porte rouge,porte,14
335  DATA ,SETC,0
336  DATA un module,module,0
337  DATA des medicaments,medicaments,0
338  DATA ,coordonnees,0
339  DATA appareils radio,radio,3
340  DATA un ticket magnetique,ticket,0
341  DATA rien de special,cloison,16
342  DATA ,*SD*,0
343  DATA ,VIEW,0
```

```
344  DATA "rien de special",lit,30
345  DATA ,STAT,0
346  DATA ,SELO,0
347  DATA ,transmetteur,0
348  DATA rien de special,controle,6
349  DATA ,code,0
350 DATA petit paquet,paquet,15
500  REM ************ DESCRIPTIONS DE L
IEUX
501  DATA dans la centrale principale,3,
3,16,0,0,0
502  DATA dans la salle de transmissions
,6,0,0,0,0,0
503  DATA dans la centrale radio,1,1,0,0
,0,0
504  DATA au pupitre de pilotage,3,0,0,6
,0,0
505  DATA ,0,0,0,0,0,0
506  DATA devant un controle d'acces,0,2
,0,0,0,0
507  DATA dans un entrepont,0,0,23,0,0,0
508  DATA dans la division d'astronomie,
0,0,22,0,0,0
509  DATA dans la coupole superieure,0,0
,0,0,0,22
510  DATA dans la station Medo,0,0,0,22,
0,0
511  DATA dans la cantine,0,0,30,24,0,0
512  DATA dans la salle de transmissions
,12,12,18,13,0,0
513  DATA dans une salle de fret,0,0,12,
0,0,0
514  DATA dans la salle des machines,0,0
,17,0,0,0
515  DATA dans la halle de depot,0,2,0,0
,0,0
516  DATA devant une cloison de securite
,0,0,24,0,0,0
```

```
517  DATA dans un passage,19,19,26,14,0,
0
518  DATA dans un passage,18,18,27,12,0,
0
519  DATA dans un passage,17,17,20,0,0,0
520  DATA dans le depot de pieces detach
ees,0,0,0,19,0,0
521  DATA dans un entrepont,0,0,25,0,0,0
522  DATA dans le sas antigravitationnel
,0,0,10,8,9,23
523  DATA dans le sas antigravitationnel
,0,0,0,0,22,24
524  DATA dans le sas antigravitationnel
,0,0,11,16,23,0
525  DATA dans le sas antigravitationnel
,0,0,0,0,24,26
526  DATA dans le sas antigravitationnel
,0,0,0,17,25,27
527  DATA dans le sas antigravitationnel
,0,0,0,18,26,0
528 DATA ,,,,,,,,,,,,,,
530  DATA dans une salle d'equipage,0,0,
0,11,0,0
600  REM ******************** MESSAGES
601  m$(1)="Je ne vois rien de special."
602  m$(2)="Je ne suis pas si fort !"
603  m$(3)="Comment vois-tu cela ?"
604 m$(4)="Mais je l'ai deja !"
605  m$(5)="Je ne comprends pas ce que t
u veux dire !"
690 ok$="D'accord !"
699  REM ************ ici eventuellement
2eme titre
800  FOR i=1 TO av
810  READ verb$(i):verb$(i)=LEFT$(verb$(
i),wl):verb$(i)=UPPER$(verb$(i))
820  NEXT i
830  FOR objet=1 TO ao
```

```
840  READ ob$(objet),nom$(objet),ob(obje
t):nom$(objet)=LEFT$(nom$(objet),wl):nom
$(objet)=UPPER$(nom$(objet)):NEXT objet
850  FOR lieu=1 TO ar:READ lieu$(lieu)
860  FOR direction=1 TO 6:READ passage(l
ieu,direction)
870  NEXT direction:NEXT lieu
880  PRINT:INPUT"   Voulez-vous des cons
eils";entree$:entree$=UPPER$(entree$)
890  IF LEFT$(entree$,1)="O"THEN GOSUB 9
00:GOTO 1000 ELSE GOTO 1000
900  MODE 1:PAPER 0:INK 0,0:BORDER 0:PEN
 1:INK 1,25
910 ancien=0: PRINT"       CPC - Ventures
":PRINT TAB(15)CHR$(164);" 1984  by J"CH
R$(178);"rg Walkowiak
920  PRINT STRING$(40,208):PRINT"Imagine
z un robot que vous pourriez     command
er avec de nombreux ordres."
930  PRINT"Je suis ce robot et je m'expo
serai pour vous aux aventures les plus r
isquees."
940  PRINT"Pour que vous puissiez me com
mander     utilement, je vous decrirai c
haque fois precisement la situation dans
 laquelle  je me trouve."
950  PRINT"Vous me direz ensuite avec de
ux mots     comme par exemple PRENDS COUT
EAU ce que je dois faire.":PRINT
960  PRINT"Je comprends en outre les ins
tructions  INVENTAIRE, SAVE, LOAD et END
."
970  PRINT STRING$(40,210):INK 3,12,24:I
NK 2,24,12:PEN 3: PRINT"       Appuyez";
:PEN 2:PRINT" <sur une touche>";:PEN 1:P
RINT
980  entree$=INKEY$:IF entree$="" THEN 9
80 ELSE MODE 1:INK 1,26:INK 2,1:INK 3,0:
INK 0,1:BORDER 1:RETURN
```

```
1000 INK 1,26:INK 2,1:INK 3,0:INK 0,1:BO
RDER 1:lignevide$=STRING$(40,32)
1010  DATA le Nord,le Sud,l'Ouest,l'Est,
en haut,en bas
1020  FOR direction=1 TO 6
1030  READ direction$(direction)
1040  NEXT direction
1050 WINDOW#1,1,40,1,16
1070 WINDOW#0,1,40,17,25:PAPER 2:INK 2,1
4:CLS#0
1080  PRINT:coup=coup+1:IF ancien<>joueu
r THEN GOSUB 30000
1090  LOCATE 1,1
1100  FOR ligne=1 TO 7
1110  PRINT lignevide$;
1120  NEXT ligne
1130  LOCATE 1,1:PEN 1
1140  PRINT"Je suis ";:ancien=joueur
1150  PRINT lieu$(joueur);:PRINT"."
1160  PRINT"Je vois ";:imprime=0
1170  FOR i=1 TO ao
1180  IF ob(i)<>joueur THEN 1210
1190  IF POS(#0)+LEN(ob$(i))+2<=39 THEN
PRINT ob$(i);", ";:GOTO 1205
1200  IF POS(#0)+LEN(ob$(i))+2>39 THEN P
RINT:GOTO 1190
1205  imprime=-1
1210  NEXT i
1215  IF NOT imprime THEN PRINT"rien de
special  ";
1220  PRINT CHR$(8);CHR$(8);"."
1230  PRINT lignevide$
1240  PRINT"Je peux aller vers ";:imprim
e=0
1250  FOR direction=1 TO 6
1260  IF passage(joueur,direction)=0 THE
N GOTO 1300 ELSE imprime=-1
1270  IF POS(#0)=20 THEN PRINT direction
$(direction);:GOTO 1300
```

```
1280  IF POS(#0)+LEN(direction$(directio
n))<38 THEN PRINT", ";direction$(directi
on);:GOTO 1300
1290  PRINT",":PRINT direction$(directio
n);:GOTO 1300
1300  NEXT direction
1310  IF imprime=0 THEN PRINT"nulle part
";
1320  PRINT".":PEN 3
1350 IF fl(1)=0 THEN PRINT"Alerte !";:IF
 coup>11 THEN m$(0)="Un asteroide nous a
 heurte.":GOTO 4500
1390  PEN 3:LOCATE 1,9:INPUT"Que dois-je
 faire";entree$:entree$=UPPER$(entree$)
1400  IF LEN(entree$)>2 THEN 1500
1410  IF entree$="N" AND passage(joueur,
1)<>0 THEN joueur=passage(joueur,1):PRIN
T ok$:GOTO 1080
1420  IF entree$="S" AND passage(joueur,
2)<>0 THEN joueur=passage(joueur,2):PRIN
T ok$:GOTO 1080
1430  IF entree$="O" AND passage(joueur,
3)<>0 THEN joueur=passage(joueur,3):PRIN
T ok$:GOTO 1080
1440  IF entree$="E" AND passage(joueur,
4)<>0 THEN joueur=passage(joueur,4):PRIN
T ok$:GOTO 1080
1450  IF entree$="H" AND passage(joueur,
5)<>0 THEN joueur=passage(joueur,5):PRIN
T ok$:GOTO 1080
1460  IF entree$="B" AND passage(joueur,
6)<>0 THEN joueur=passage(joueur,6):PRIN
T ok$:GOTO 1080
1470  IF LEN(entree$)<3 THEN PRINT"Aucun
 chemin n'y mene !":GOTO 1080
1480  IF LEN(entree$)>8 THEN GOTO 2000
1499  REM ********** DEBUT INVENTA
IRE
1500  IF LEFT$(entree$,3)<>"INV" THEN GO
TO 1600
```

```
1510  PRINT"Voici ce que je porte sur mo
i:"
1520  FOR objet=1 TO ao
1530  IF ob(objet)=-1 THEN PRINT ob$(obj
et)
1540  NEXT objet
1550  GOTO 1080
1560  REM ********** FIN INVENTAIR
E
1600  IF LEFT$(entree$,3)<>"SAV" THEN GO
TO 1700
1610  PRINT"Appuyer sur REC & PLAY ...
            Sauvegarder sous quel nom ..
.";STRING$(4,8);:INPUT entree$
1620  IF LEN(entree$)>10 THEN PRINT"Un p
eu plus court S.V.P !":GOTO 1610 ELSE en
tree$="!"+entree$+".jeu":OPENOUT entree$
1630  PRINT#9,joueur
1640  FOR objet=1 TO ao
1650  PRINT#9,ob(objet)
1660  NEXT objet
1670  FOR lieu=1 TO ar:FOR direction=1 T
O 6:PRINT#9,passage(lieu,direction):NEXT
 direction:NEXT lieu
1680  FOR flag=1 TO af:PRINT#9,fl(flag):
NEXT flag
1690  CLOSEOUT:PRINT ok$:GOTO 1080
1700  IF LEFT$(entree$,3)<>"LOA" THEN GO
TO 1900
1710  PRINT"Rembobiner la cassette et ap
puyer sur   PLAY ...
            Quel jeu faut-il charger ...
";STRING$(4,8);:INPUT entree$
1720  IF LEN(entree$)>10 THEN PRINT"Ce n
'est pas possible !":GOTO 1710 ELSE entr
ee$="!"+entree$+".JEU":OPENIN entree$
1730  INPUT#9,joueur
1740  FOR objet=1 TO ao:INPUT#9,ob(objet
):NEXT objet
```

```
1750  FOR lieu=1 TO ar:FOR direction=1 T
O 6:INPUT#9,passage(lieu,direction):NEXT
 direction:NEXT lieu
1760  FOR flag=1 TO af:INPUT#9,fl(flag):
NEXT flag
1770  CLOSEIN:PRINT ok$:GOTO 1080
1900  IF LEFT$(entree$,3)<>"INS" THEN GO
TO 1950
1910  GOSUB 900:GOTO 1070
1950  IF LEFT$(entree$,3)<>"END" THEN GO
TO 2000
1960 MODE 1:PRINT"Nous vous souhaitons p
lus de succes la  prochaine fois.":PRINT
:PRINT:PRINT:END
2000  longueur=LEN(entree$)
2010  FOR lettre=1 TO longueur
2020  tester$=MID$(entree$,lettre,1)
2030  IF tester$<>" "THEN NEXT lettre
2040  everb$=LEFT$(entree$,wl)
2050  rl=longueur-lettre
2060  IF rl<0 THEN 2090
2070  eobjet$=RIGHT$(entree$,rl)
2080  eobjet$=LEFT$(eobjet$,wl)
2090  FOR numeroverb=1 TO av
2100  IF everb$=verb$(numeroverb) THEN 2
130
2110  NEXT numeroverb
2120  PRINT"Je ne comprends pas ce verbe
.":GOTO 1080
2130  FOR o=1 TO ao
2140  IF eobjet$=nom$(o) THEN 2200
2150  NEXT o
2160  PRINT"Je ne connais pas cet objet.
":GOTO 1080
2200  ON numeroverb GOTO 5000, 2210, 700
0, 8000, 9000, 10000, 11000, 12000, 1300
0, 14000
2210 nombre=0:FOR i=1 TO ao
2220 IF ob(i)=-1 THEN nombre=nombre+1
2230 IF nombre=imax THEN PRINT"Non merci
 - Je porte deja assez de      choses.":
GOTO 1080
```

```
2240 NEXT i
2250 GOTO 6000
4500 MODE 1:CLS:REM JOUEUR MORT
4510  PRINT"Rien ne m'aura ete epargne !
":PRINT:PRINT m$(0)
4520  PRINT:PRINT"Je suis mort !":PRINT
4530  INPUT"Dois-je essayer encore une f
ois ";entree$:entree$=UPPER$(entree$)
4540  IF LEFT$(entree$,1)="O" THEN RUN
4550  GOTO 1960
4800 MODE 1:REM JOUEUR GAGNE
4810  PRINT"Toutes nos felicitations !"
4815 PRINT:PRINT"Vous avez atteint Pia 3
 a temps, et vousavez ainsi accompli l'e
preuve qui vous  etait imposee. Vous pou
vez maintenant   tenter une autre aventu
re."
4830  PRINT:PRINT:PRINT:PRINT:END
5000 IF ob(o)<>joueur AND ob(o)<>-1 THEN
 GOTO 5900
5005 IF o=1 THEN PRINT"Il me sert a prog
rammer le pilote       automatique.":ob(
22)=1:GOTO 1080
5006 IF o=2 AND fl(1)=0 THEN PRINT"L'ecr
an montre un asteroide.":GOTO 1080
5007 IF o=2 AND fl(1)=-1 THEN PRINT"Je n
e vois rien de special.":GOTO 1080
5009 IF o=40 THEN GOTO 51500
5010 IF o=6 THEN PRINT m$(1):GOTO 1080
5013 IF o=25 AND ob(40)=0 THEN PRINT"Des
sous se trouvait un ticket          mag
netique.":ob(40)=11:ancien=0:GOSUB 30000
:GOTO 1080
5014 IF o=8 AND joueur=23 AND fl(7)=0 TH
EN PRINT"Elle est fermee.":GOTO 1080
5015 IF o=8 AND joueur=25 AND fl(8)=0 TH
EN PRINT"Elle est fermee.":GOTO 1080
5016 IF o=8 AND joueur=23 AND fl(7)=-1 T
HEN PRINT"Elle mene a un entrepont.":GOT
O 1080
```

```
5017 IF o=8 AND joueur=25 AND fl(8)=-1 T
HEN PRINT"Elle mene a un entrepont.":GOT
O 1080
5019 IF o=39 THEN PRINT m$(1):ob(5)=joue
ur:ob(6)=joueur:ob(39)=0:GOTO 1080
5020 IF o=5 THEN GOTO 20000
5021 IF o=44 AND ob(29)=0 THEN PRINT"Mon
 scaphandre est dessous.":GOTO 1080
5025 IF o=4 THEN PRINT"On dispose des ch
iffres de 0 a 9.":GOTO 1080
5026 IF o=10 AND ob(32)<>-1 THEN PRINT"J
e ne peux pas l'ouvrir.":GOTO 1080
5027 IF o=10 AND ob(32)=-1 THEN PRINT"De
dans se trouve l'ordinateur central.":ob
(11)=7:ob(10)=0:GOTO 1080
5028 IF o=11 THEN PRINT"Je vois de nombr
eux modules.":ob(13)=joueur:ob(15)=0:GOT
O 1080
5029 IF o=13 THEN PRINT"Ce sont des modu
les enfichables.":GOTO 1080 b(15)=0:GOTO
 1080
5030 IF o=12 THEN PRINT"... un module in
formatique normal":GOTO 1080
5031 IF o=14 THEN PRINT m$(1):GOTO 1080
5032 IF o=15 THEN PRINT m$(1):GOTO 1080
5033 IF o=16 THEN PRINT"Ils font partie
du systeme d'aeration.":GOTO 1080
5034 IF o=17 THEN PRINT"Ce n'est qu'une
indication de chemin.":GOTO 1080
5035 IF o=20 THEN PRINT m$(1):GOTO 1080
5036 IF o=33 THEN PRINT m$(1):ob(30)=jou
eur:GOTO 1080
5037 IF o=22 THEN PRINT"Je vois des touc
hes: LAND MOVE STOP &   SETC.":GOTO 1080
5038 IF o=23 THEN PRINT m$(1):GOTO 1080
5039 IF o=24 AND ob(32)=0 THEN PRINT"Der
riere se trouve un tournevis.":GOTO 1080
5040 IF o=32 THEN PRINT m$(1):GOTO 1080
5041 IF o=27 THEN PRINT m$(1):GOTO 1080
5042 IF o=28 AND fl(4)=0 THEN PRINT"Elle
 est vide.":GOTO 1080
```

```
5043 IF o=28 AND fl(4)=-1 THEN PRINT"Ell
e est pleine.":GOTO 1080
5044 IF o=29 THEN PRINT"C'est le mien.":
GOTO 1080
5045 IF o=30 THEN PRINT m$(1):GOTO 1080
5046 IF o=48 THEN PRINT m$(1):GOTO 1080
5200 IF o=24 THEN PRINT m$(1):GOTO 1080
5899 PRINT m$(1):GOTO 1080
5900 IF o=7 AND joueur=24 THEN PRINT m$(
1):GOTO 1080
5902 IF o=7 AND joueur=23 THEN fl(5)=-1:
ancien=0:ob(8)=joueur:GOTO 1080
5903 IF o=7 AND joueur=24 THEN PRINT m$(
1):GOTO 1080
5904 IF o=7 AND joueur=25 THEN fl(6)=-1:
ob(9)=joueur:ancien=1:GOTO 1080
5905 IF o=7 AND joueur=26 THEN PRINT m$(
1):GOTO 1080
5906 IF o=7 AND joueur=27 THEN PRINT m$(
1):GOTO 1080
5907 IF o=7 AND joueur=22 THEN PRINT m$(
1):GOTO 1080
5910 IF o=20 AND joueur=8 THEN PRINT"PIA
 III est indique par 151064.":GOTO 1080
5916 IF o=25 AND ob(40)=-1 THEN GOTO 108
0
5920 IF o=32 THEN PRINT m$(1):GOTO 1080
5922 IF o=8 AND joueur=25 AND passage(25
,4)=0 THEN PRINT"Elle est fermee.":GOTO
1080
5923 IF o=47 AND joueur=2 THEN PRINT"Le
meme modele.":GOTO 1080
5924 IF o=4 AND joueur=2 THEN PRINT"Le m
eme modele.":GOTO 1080
5925 IF o=20 AND joueur=15 THEN PRINT"Ce
 sont les medicaments.":GOTO 1080
5990 PRINT"Je ne vois rien de tel ici.":
GOTO 1080
6000 IF ob(o)<>joueur AND ob(o)<>-1 THEN
 GOTO 6900
```

```
6005 IF o=40 THEN PRINT ok$:ob(o)=-1:anc
ien=0:GOSUB 30000:GOTO 1080
6010 IF o=12 AND ob(o)=-1 THEN PRINT m$(
4):GOTO 1080
6011 IF o=12 THEN ob(12)=-1:PRINT ok$:GO
TO 1080
6012 IF o=13 THEN PRINT"J'ai un module."
:ob(36)=-1:GOTO 1080
6013 IF o=14 OR o=28 THEN PRINT ok$:ob(o
)=-1:GOTO 1080
6014 IF (o=30 OR o=50) THEN PRINT ok$:ob
(o)=-1:ancien=0:GOTO 1080
6800 IF o=32 THEN PRINT ok$:ob(o)=-1:GOT
O 1080
6890 IF o=joueur THEN PRINT m$(2):GOTO 1
080
6900 IF o=29 THEN PRINT ok$:ob(o)=-1:GOT
O 1080
6915 IF o=20 AND joueur=15 THEN PRINT ok
$:ob(50)=-1:GOTO 1080
6920 IF o=32 AND ob(o)=0 THEN ob(o)=-1:P
RINT ok$:GOTO 1080
6990 PRINT"Je ne vois rien de tel ici.":
GOTO 1080
7000 IF o=41 AND ob(40)=-1 THEN PRINT ok
$:passage(16,4)=1:ancien=2:GOTO 1080
7005 IF o=41  AND ob(40)<>-1 THEN PRINT"
Ce n'est pas possible pour le moment.":G
OTO 1080
7010 IF o=8 AND ob(14)<>-1 THEN PRINT"Je
 n'ai rien pour ouvrir.":GOTO 1080
7011 IF o=8 AND joueur=23 AND ob(14)=-1
THEN PRINT ok$:passage(23,4)=7:fl(7)=-1:
GOTO 1080
7012 IF o=8 AND joueur=25 AND ob(14)=-1
THEN PRINT ok$:passage(25,4)=21:fl(8)=-1
:GOTO 1080
7015 IF o=23 AND joueur=9 THEN m$(0)="Un
e fuite d'oxygene m'entraine dans     l'
espace.":GOTO 4500
```

```
7020 IF o=8 AND joueur=14 THEN m$(0)="J'
ai ete atteint par une decharge de    ra
yons du reacteur.":GOTO 4500
7099 PRINT m$(5):GOTO 1080
8000 IF o=46 AND fl(2)=-1 THEN PRINT"Cli
c !":passage(24,6)=25:GOTO 1080
8012 IF o=46 AND fl(2)=0 THEN PRINT"Rien
 ne se passe.":GOTO 1080
8020 IF o=45 THEN GOTO 20200
8021 IF o=43 THEN GOTO 20300
8022 IF o=42 THEN m$(0)="C'etait un suic
ide.":GOTO 4500
8024 IF o=18 THEN m$(0)="Le pilote autom
atique a pose le vaisseausur le soleil."
:GOTO 4500
8025 IF o=26 AND tm$="151063" THEN m$(0)
="Mauvaise fonction - fausses coordonnee
s.":GOTO 4500
8026 IF o=26 AND fl(1)=0 THEN PRINT"Le v
aisseau evite l'asteroide.":fl(1)=-1:GOT
O 1080
8027 IF o=26 AND fl(1)=-1 AND fl(10)=0 T
HEN m$(0)="Pas de cap fixe - le but a et
e un       soleil.":GOTO 4500
8028 IF o=26 AND fl(10)=-1 AND ob(50)<>1
3 THEN m$(0)="Les habitants de PIA III m
'ont lynche   parce que je suis arrive s
ans les       medicaments.":GOTO 4500
8029 IF o=31 THEN fl(1)=0:PRINT ok$:GOTO
 1080
8030 IF o=35 THEN INPUT"Coordonnees ";tm
$:IF tm$="151064" THEN fl(10)=-1:GOTO 10
80
8031 IF o=35 THEN GOTO 1080
8032 IF o=26 AND fl(10)=-1 AND ob(50)=13
 THEN GOTO 4800
8115 IF o=23 AND joueur=9 THEN m$(0)="Un
e fuite d'oxygene m'a entraine dans   l'
espace.":GOTO 4500
```

```
8990 PRINT"Je ne vois pas ici de tel bou
ton.":GOTO 1080
9000 IF ob(o)<>-1 THEN GOTO 9900
9005 IF o=50 AND ob(50)=-1 AND joueur=13
 THEN PRINT ok$:ob(50)=joueur:ancien=0:f
l(11)=-1:GOTO 1080
9006 IF o=50 AND ob(50)=-1 AND joueur=15
 THEN PRINT ok$:ob(50)=joueur:ancien=0:f
l(11)=-1:GOTO 1080
9010 PRINT ok$:ob(o)=joueur:GOTO 1080
9900 IF o=13 AND ob(36)=-1 THEN ob(36)=j
oueur:PRINT ok$:GOTO 1080
10000 IF o=29 THEN PRINT"J'ai mis le sca
phandre.":ob(o)=-2:ob(14)=-1:GOTO 1080
10010 GOTO 6000
11000 IF o=28 AND joueur<>21 THEN PRINT"
Avec quoi ?":GOTO 1080
11010 IF o=28 AND joueur=21 AND ob(30)<>
-1 THEN PRINT"Cela ne marche pas pour le
 moment.":GOTO 1080
11020 IF o=28 AND joueur=21 AND ob(30)=-
1 THEN PRINT ok$:fl(4)=-1:GOTO 1080
11900 PRINT m$(5):GOTO 1080
12000 IF o=38 AND joueur=12 THEN INPUT t
m$:GOTO 1080
12005 IF o=38 AND joueur=2 THEN INPUT tm
$:GOTO 1080
12010 IF o=49 AND joueur=6 THEN INPUT tm
$:IF tm$="220559" THEN passage(2,1)=15:j
oueur=15:GOTO 1080
12020 IF o=49 AND joueur=6 AND tm$<>"220
559" THEN m$(0)="Les systemes de defense
 se declenchent.":GOTO 4500
13000 IF o=47 AND NOT(joueur=12 OR joueu
r=2) THEN PRINT"Je ne vois pas de transm
etteur.":GOTO 1080
13010 IF o=47 AND joueur=12 AND tm$<>"64
4664" THEN m$(0)="Mes atomes derivent da
ns l'espace.":GOTO 4500
```

```
13011 IF o=47 AND joueur=2 AND tm$<>"644
663" THEN m$(0)="Mes atomes derivent dan
s l'espace.":GOTO 4500
13020 IF o=47 AND (joueur=12 OR joueur=2
) AND tm$="644664" THEN joueur=2:GOTO 10
80
13021 IF o=47 AND (joueur=12 OR joueur=2
) AND tm$="644663" THEN joueur=12:GOTO 1
080
13030 IF o=22 AND joueur=1 THEN fl(1)=-1
:PRINT ok$:GOTO 1080
13900 PRINT m$(5):GOTO 1080
14000 IF o=13 AND joueur=7 AND ob(13)=7
THEN PRINT ok$:ob(12)=0:ob(36)=-1:fl(2)=
-1:GOTO 1080
20000 WINDOW SWAP 0,1:CLS:PRINT"Transmis
sion received:":PRINT
20010 PRINT"Terre a vaisseau spatial CC
464:":PRINT
20020 PRINT"Accident atomique sur PIA II
I"
20030 PRINT"Medicament Cibarad 92 necess
aire        d'urgence. Voler vers statio
n d'approvi-sionnement 89011.":PRINT:PRI
NT"Coordonnees:"
20040 PRINT"89011  : 64-46-64":PRINT"
      Code: 220559":PRINT"PIA III: 15-10
-63":PRINT"STATUS: **Mauvaise fonction**
"
20050 entree$=INKEY$:IF entree$="" THEN
20050 ELSE WINDOW SWAP 1,0:ancien=0:GOTO
 1080
20200 WINDOW SWAP 0,1:CLS:PRINT"Status C
heck:":PRINT
20210 IF fl(1)=0 THEN PRINT"Autopilot OF
F" ELSE PRINT"Autopilot ON"
20220 IF fl(2)=0 THEN PRINT"Unite centra
le en panne" ELSE PRINT"All Systems Go"
20230 IF fl(1)=0 THEN PRINT:PRINT"ATTENT
ION !":PRINT"Risque de collision !"
```

```
20240 IF fl(2)=-1 THEN PRINT"Situation:
644663"
20250 entree$=INKEY$:IF entree$="" THEN
20250 ELSE ancien=0:WINDOW SWAP 1,0:GOTO
 1080
21300 PLOT 320, 180:DRAW 320, 230:DRAW 4
00, 230:DRAW 400, 180:DRAW 240, 180:DRAW
 240, 230:DRAW 320, 230:PLOT 240, 230:DR
AW 260, 250:PLOT 260, 250:DRAW 380, 250:
DRAW 400, 230:PLOT 370, 250:DRAW 370, 26
0:DRAW 270, 260
21301 DRAW 270, 250:PLOT 250, 260:DRAW 2
50, 320:DRAW 390, 320:DRAW 390, 260:DRAW
 250, 260:PLOT 0, 190:DRAW 240, 190:PLOT
 400, 190:DRAW 480, 190:PLOT 480, 400:DR
AW 480, 190:DRAW 550, 120:PLOT 520, 400:
DRAW 550, 370
21302 DRAW 550, 120:DRAW 560, 120:PLOT 5
50, 370:DRAW 640, 280:PLOT 560, 360:DRAW
 560, 120:PLOT 330, 220:DRAW 390, 220:DR
AW 390, 190:DRAW 330, 190:DRAW 330, 220:
PLOT 310, 220:DRAW 250, 220:DRAW 250, 19
0:DRAW 310, 190
21303 DRAW 310, 220:PLOT 257, 237:DRAW 3
14, 237:DRAW 314, 246:DRAW 266, 246:DRAW
 257, 237:PLOT 326, 237:DRAW 383, 237:DR
AW 374, 246:DRAW 326, 246:DRAW 326, 237:
PLOT 332, 243:PLOT 338, 243:PLOT 347, 24
3:DRAW 347, 240
21304 PLOT 356, 240:DRAW 356, 243:PLOT 3
65, 243:DRAW 365, 240:PLOT 309, 243:PLOT
 309, 240:PLOT 299, 240:PLOT 292, 240:PL
OT 287, 240:PLOT 277, 240:DRAW 277, 244
21320 RETURN
21700 PLOT 0, 380:DRAW 70, 330:DRAW 70,
190:PLOT 10, 144:DRAW 70, 190:PLOT 50, 1
44:DRAW 90, 184:DRAW 90, 404:PLOT 90, 18
4:DRAW 190, 184:PLOT 190, 154:DRAW 190,
224:DRAW 450, 224:DRAW 450, 154:DRAW 190
, 154:PLOT 190, 224
20300 IF fl(i):-1 THEN PRINT m$(1): GOTO
1080 : ELSE PRINT"
```

```
21701 DRAW 200, 244:DRAW 420, 244:DRAW 4
50, 224:PLOT 440, 164:PLOT 440, 214:PLOT
 210, 214:PLOT 210, 164:PLOT 214, 240:DR
AW 414, 240:PLOT 420, 236:DRAW 212, 236:
PLOT 210, 232:DRAW 426, 232:PLOT 398, 24
4:DRAW 398, 312
21702 DRAW 402, 316:DRAW 642, 316:PLOT 6
42, 320:DRAW 398, 320:DRAW 390, 312:DRAW
 390, 244:PLOT 450, 180:DRAW 642, 180:PL
OT 192, 232:DRAW 160, 232:DRAW 156, 228:
DRAW 156, 180:PLOT 44, 312:DRAW 20, 312:
PLOT 20, 284
21703 DRAW 44, 284:PLOT 20, 256:DRAW 44,
 256:PLOT 44, 340:DRAW 20, 340:PLOT 20,
228:DRAW 44, 228:PLOT 44, 200:DRAW 20, 2
00:PLOT 20, 172:DRAW 44, 172
21720 RETURN
21800 PLOT 0, 399:DRAW 639, 399:PLOT 639
, 359:DRAW 359, 359:DRAW 359, 159:DRAW 6
39, 159:PLOT 639, 319:DRAW 359, 319:PLOT
 399, 279:PLOT 439, 239:PLOT 519, 239:PL
OT 519, 279:PLOT 479, 279:PLOT 399, 199:
PLOT 404, 204
21801 PLOT 374, 234:PLOT 429, 289:PLOT 4
74, 244:PLOT 469, 199:PLOT 554, 189:PLOT
 589, 224:PLOT 619, 269:PLOT 589, 299:PL
OT 569, 314:PLOT 569, 269:PLOT 289, 359:
DRAW 79, 359:DRAW 79, 279:DRAW 289, 279:
DRAW 289, 359:RETURN
22000 PLOT-5, 188:DRAW 465, 188:DRAW 495
, 158:DRAW 495, 338:DRAW 575, 258:DRAW 5
75, 148:PLOT 495, 158:DRAW 505, 158:DRAW
 505, 328:PLOT 465, 188:DRAW 465, 398:PL
OT 420, 310:DRAW 400, 310:DRAW 400, 330:
DRAW 380, 330
22001 DRAW 380, 310:DRAW 360, 310:DRAW 3
60, 290:DRAW 380, 290:DRAW 380, 270:DRAW
 400, 270:DRAW 400, 290:DRAW 420, 290:DR
AW 420, 310:PLOT 320, 240:DRAW 350, 210:
DRAW 150, 210:DRAW 120, 240:DRAW 320, 24
0:PLOT 0, 250
```

```
22002 DRAW 50, 250:DRAW 50, 340:DRAW-10,
 340:PLOT 50, 340:DRAW 30, 360:DRAW 0, 3
60:PLOT 40, 300:DRAW 40, 290:RETURN
22100 PLOT 75, 189:DRAW 75, 409:PLOT 75,
 189:PLOT 33, 147:DRAW 74, 188:PLOT 234,
 188:DRAW 414, 188:DRAW 389, 248:PLOT 23
4, 188:DRAW 259, 248:DRAW 389, 248:PLOT
75, 238:DRAW 255, 238:PLOT 395, 238:DRAW
 525, 238
22101 DRAW 525, 398:PLOT 525, 238:DRAW 5
85, 178:DRAW 585, 398:PLOT 585, 178:DRAW
 595, 178:DRAW 595, 398:PLOT 385, 188:DR
AW 385, 148:DRAW 375, 148:DRAW 375, 188:
PLOT 265, 188:DRAW 265, 148:DRAW 275, 14
8:DRAW 275, 188
22105 WINDOW SWAP 0,1:LOCATE 10,4:PRINT"
 La terre est loin -";:LOCATE 10,5:PRINT
"Mac Donald's aussi !":IF ob(40)=11 THEN
 LOCATE 19,12:PRINT CHR$(131);
22106 WINDOW SWAP 1,0
22110 RETURN
22200 PLOT 50, 398:DRAW 50, 228:DRAW 0,
178:PLOT 120, 398:DRAW 120, 298:DRAW 220
, 398:PLOT 120, 298:DRAW 110, 298:DRAW 1
10, 398:PLOT 420, 248:DRAW 420, 278:DRAW
 460, 278:DRAW 460, 248:DRAW 420, 248:PL
OT 420, 278
22201 DRAW 430, 288:DRAW 450, 288:DRAW 4
60, 278:PLOT 410, 240:DRAW 470, 240:DRAW
 480, 220:DRAW 400, 220:DRAW 410, 240:PL
OT 414, 235:PLOT 418, 235:PLOT 422, 235:
PLOT 426, 235:PLOT 430, 235:PLOT 434, 23
5:PLOT 438, 235
22202 PLOT 442, 235:PLOT 446, 235:PLOT 4
50, 235:PLOT 458, 235:PLOT 462, 235:PLOT
 466, 235:PLOT 468, 231:PLOT 465, 231:PL
OT 460, 230:PLOT 462, 226:PLOT 466, 226:
PLOT 470, 226:PLOT 453, 229:DRAW 412, 22
9:PLOT 430, 248
```

```
22203 DRAW 430, 240:PLOT 452, 240:DRAW 4
52, 248:PLOT 402, 218:DRAW 422, 198:DRAW
 422, 168:DRAW 462, 168:DRAW 462, 198:DR
AW 482, 218:PLOT 462, 198:DRAW 422, 198:
PLOT 362, 398:DRAW 362, 318:DRAW 382, 29
8:DRAW 472, 298
22204 DRAW 492, 318:DRAW 492, 398:PLOT 4
22, 338:DRAW 382, 298:DRAW 362, 318:DRAW
 422, 338:PLOT 472, 298:DRAW 432, 338:DR
AW 492, 318:PLOT 432, 338:DRAW 422, 338:
PLOT 422, 398:DRAW 422, 338:DRAW 432, 33
8:DRAW 432, 398
22205 PLOT 332, 284:DRAW 426, 284:PLOT 4
54, 284:DRAW 534, 284:DRAW 554, 304:DRAW
 524, 364:DRAW 494, 394:PLOT 332, 284:DR
AW 312, 304:DRAW 332, 364:DRAW 362, 394:
PLOT 332, 284:DRAW 332, 264:DRAW 421, 26
4:PLOT 461, 264
22206 DRAW 535, 264:DRAW 535, 285:PLOT 5
35, 264:DRAW 555, 284:DRAW 555, 304:PLOT
 333, 264:DRAW 313, 284:DRAW 313, 304:PL
OT 593, 404:DRAW 643, 354
22220 RETURN
22300 PLOT 0, 390:DRAW 130, 320:DRAW 530
, 320:DRAW 640, 390:PLOT 530, 320:DRAW 5
30, 200:PLOT 620, 145:DRAW 531, 199:DRAW
 131, 199:DRAW 131, 319:PLOT 131, 199:DR
AW 91, 179:DRAW 91, 299:DRAW 1, 339:PLOT
-9, 179
22301 IF ob(50)=joueur THEN WINDOW SWAP
0,1:LOCATE 13,13:PRINT STRING$(8,232):WI
NDOW SWAP 1,0
22320 RETURN
22400 PLOT 120, 400:DRAW 120, 240:DRAW 1
90, 310:PLOT 190, 400:DRAW 190, 270:DRAW
 320, 270:DRAW 320, 350:DRAW 300, 370:DR
AW 250, 370:DRAW 220, 400:PLOT 250, 370:
DRAW 290, 410:PLOT 300, 370:DRAW 330, 40
0:PLOT 320, 350
```

```
22401 DRAW 370, 400:PLOT 320, 270:DRAW 4
50, 400:PLOT 420, 400:DRAW 330, 310:DRAW
 330, 290:DRAW 440, 400:PLOT 310, 280:DR
AW 200, 280:DRAW 200, 300:DRAW 310, 300:
DRAW 310, 280:PLOT 590, 400:DRAW 620, 37
0:DRAW 620, 400
22402 DRAW 623, 370:DRAW 623, 400:PLOT 6
23, 370:DRAW 638, 355
22420 RETURN
22600 PLOT 160, 144:DRAW 220, 184:DRAW 4
60, 184:DRAW 520, 144:PLOT 460, 184:DRAW
 460, 384:DRAW 500, 404:PLOT 460, 384:DR
AW 220, 384:DRAW 160, 404:PLOT 220, 384:
DRAW 220, 184:PLOT 280, 189:DRAW 410, 18
9:DRAW 415, 194
22601 DRAW 415, 369:DRAW 410, 374:DRAW 2
75, 374:DRAW 270, 369:PLOT 270, 369:DRAW
 270, 194:DRAW 275, 189:DRAW 280, 189:PL
OT 425, 284:DRAW 425, 274:DRAW 430, 274:
DRAW 430, 284:DRAW 425, 284:PLOT 428, 28
0:PLOT 428, 279
22610 IF ((joueur=16 AND passage(16,4)<>
1) OR joueur=9 OR joueur=6) THEN PLOT 34
3,188:DRAW 343,375:GOTO 22613
22611 PLOT 274, 241:DRAW 329, 241:DRAW 3
29, 266:DRAW 272, 266:PLOT 328, 266:DRAW
 322, 272:DRAW 270, 272:DRAW 270, 276
22612 DRAW 318, 276:DRAW 318, 272:PLOT 3
18, 276:DRAW 322, 276:DRAW 322, 304:DRAW
 270, 304:PLOT 328, 250:DRAW 350, 250:DR
AW 350, 374:PLOT 350, 250:DRAW 396, 226:
DRAW 396, 374
22613 RETURN
22700 PLOT 60, 398:DRAW 60, 198:DRAW 10,
 148:PLOT 60, 248:DRAW 260, 248:PLOT 260
, 398:DRAW 260, 248:DRAW 360, 348:PLOT 3
60, 398:DRAW 360, 248:DRAW 480, 248:PLOT
 480, 398:DRAW 480, 188:DRAW 640, 188
22720 RETURN
```

```
23000 PLOT 0, 190:DRAW 430, 190:PLOT 430
, 400:DRAW 430, 190:PLOT 430, 191:DRAW 4
75, 146:PLOT 493, 146:DRAW 493, 362:DRAW
 583, 272:DRAW 583, 144:PLOT 498, 355:DR
AW 498, 145
23020 RETURN
23100 PLOT 120, 400:DRAW 120, 230:DRAW 3
3, 143:PLOT-3, 170:DRAW 66, 239:DRAW 60,
 239:DRAW-3, 176:PLOT 60, 239:DRAW 60, 3
98:PLOT 66, 398:DRAW 66, 239:PLOT 32, 27
3:DRAW 2, 273:PLOT 2, 298:DRAW 32, 298:P
LOT 32, 323
23101 DRAW 2, 323:PLOT 2, 348:DRAW 32, 3
48:PLOT 32, 373:DRAW 2, 373:PLOT 2, 398:
DRAW 32, 398:PLOT 32, 248:DRAW 2, 248:PL
OT 2, 223:DRAW 32, 223:PLOT 2, 198:DRAW
17, 198:PLOT 267, 208:DRAW 267, 178:DRAW
 497, 208
23102 DRAW 497, 178:DRAW 267, 208:PLOT 2
57, 208:DRAW 507, 208:DRAW 517, 218:DRAW
 517, 308:DRAW 507, 318:DRAW 257, 318:DR
AW 247, 308:DRAW 247, 218:DRAW 257, 208:
PLOT 267, 218:DRAW 497, 218:DRAW 497, 30
8:DRAW 267, 308
23103 DRAW 267, 218:PLOT 247, 248:DRAW 2
27, 248:PLOT 227, 246:DRAW 247, 246:PLOT
 234, 244:DRAW 234, 238:DRAW 234, 254:PL
OT 230, 254:DRAW 238, 254:PLOT 121, 230:
DRAW 247, 230:PLOT 516, 230:DRAW 636, 23
0:PLOT 519, 219
23104 DRAW 519, 225:PLOT 426, 318:DRAW 4
26, 366:DRAW 477, 366:DRAW 480, 369:DRAW
 426, 369:DRAW 423, 366:DRAW 423, 321:PL
OT 396, 321:DRAW 396, 327:DRAW 393, 327:
DRAW 393, 321
23120 RETURN
23200 GOSUB 31200:GOSUB 31400:GOSUB 3130
0
```

```
23300 GOSUB 31200:IF fl(5) AND joueur=23
 THEN GOSUB 31400
23305 IF fl(6) AND joueur=25 THEN GOSUB
31400
23310 RETURN
23400 GOSUB 31200:GOSUB 31300:GOSUB 3140
0:RETURN
23600 GOSUB 31200:GOSUB 31400:RETURN
23700 GOSUB 31200:GOSUB 31400:RETURN
24000 PLOT 0, 160:DRAW 60, 160:DRAW 60,
190:DRAW 0, 190:PLOT 60, 160:DRAW 120, 2
20:PLOT 120, 220:DRAW 120, 250
24001 DRAW 60, 190:PLOT 120, 250:DRAW 0,
 250:PLOT 120, 230:DRAW 460, 230:PLOT 54
6, 144:DRAW 460, 230:DRAW 460, 400:PLOT
490, 370:DRAW 490, 220:DRAW 500, 220:DRA
W 500, 360:PLOT 490, 370:DRAW 570, 290:D
RAW 570, 150
24002 PLOT 490, 220:PLOT 566, 144:DRAW 4
90, 220:PLOT 500, 220:DRAW 571, 150:DRAW
 501, 220:PLOT 421, 300:DRAW 411, 300:DR
AW 412, 299:DRAW 419, 299:DRAW 419, 302:
DRAW 413, 302:DRAW 414, 304:DRAW 416, 30
4:DRAW 416, 297
24003 DRAW 415, 297:RETURN
30000 PAPER 0:CLS#1:IF joueur>20 THEN 30
030
30005 IF joueur>10 THEN 30020
30010 ON joueur GOSUB 21300,22200,21300,
21400,21500,22600,21700,21800,22600,2200
0
30011 PAPER 2:RETURN
30020 ON joueur-10 GOSUB 22100,22200,223
00,22400,22300,22600,22700,22700,22700,2
3000
30021 PAPER 2:RETURN
30030 ON joueur-20 GOSUB 23100,23200,233
00,23400,23300,23600,23700,22600,23900,2
4000
30031 PAPER 2:RETURN
```

```
31200 PLOT 180, 400:PLOT 180, 144:DRAW 1
80, 414:PLOT 450, 144:DRAW 450, 414:PLOT
 210, 354:DRAW 270, 354:PLOT 360, 354:DR
AW 420, 354:PLOT 420, 294:DRAW 360, 294:
PLOT 270, 294:DRAW 210, 294:PLOT 210, 23
4
31201 DRAW 270, 234:PLOT 360, 234:DRAW 4
20, 234:PLOT 420, 174:DRAW 360, 174:PLOT
 270, 174:DRAW 210, 174:RETURN
31300 PLOT 360, 234:DRAW 420, 234:PLOT 4
20, 174:DRAW 360, 174:PLOT 270, 174:DRAW
 210, 174:PLOT 140, 344:DRAW 140, 184:DR
AW 90, 144:DRAW 90, 384:DRAW 140, 344:DR
AW 130, 344:DRAW 130, 184:DRAW 140, 184
31301 DRAW 130, 184:DRAW 90, 154:PLOT 13
0, 344:DRAW 90, 374:RETURN
31400 PLOT 477, 344:PLOT 477, 344:DRAW 4
77, 179:DRAW 527, 139:DRAW 527, 384:DRAW
 477, 344:PLOT 477, 344:PLOT 487, 344:DR
AW 477, 344:PLOT 527, 379:DRAW 487, 344:
DRAW 487, 179:DRAW 527, 144:PLOT 487, 17
9:DRAW 477, 179:RETURN
51500 WINDOW SWAP 0,1:CLS:PRINT CHR$(150
);STRING$(22,154);CHR$(156)
51510 PRINT CHR$(149);"  ID - Card# 435G
ER   ";CHR$(149)
51520 PRINT CHR$(149);"         pour
     ";CHR$(149)
51530 PRINT CHR$(149);"       High Res
     ";CHR$(149)
51540 PRINT CHR$(149);STRING$(22,32);CHR
$(149)
51550 PRINT CHR$(149);"   Security Level
 I   ";CHR$(149)
51560 PRINT CHR$(149);"   (c) TERRA  IX.
84   ";CHR$(149)
51570 PRINT CHR$(147);STRING$(22,154);CH
R$(153)
51580 entree$=INKEY$:IF entree$="" THEN
51580 ELSE WINDOW SWAP 1,0:ancien=0:GOTO
 1080
```

EDITEUR DE JEUX D'AVENTURE

```
5 REM Adventure Editor
6 REM (c) 1984 by J. Walkowiak
7 REM ------------------------------
10 MODE 1:CLEAR:DEFINT a-z:DIM ra$(60),o
b$(60),rn$(60),ob(60),ve$(30),m$(60),ac$
(30,60),bc$(30,60),ad$(20),du(60,6)
20 ar=0:ao=0:av=0:am=0:af=0:i2=1:l3=1:m1
$="       ADVENTURE EDITOR   Ver 1.0":m2
$=STRING$(2,10):m2$=m2$+"Que voulez-vous
":m3$=STRING$(40,154):m4$=STRING$(40,32)
30 CLS:PRINT m1$:PRINT m3$;
40 PRINT:PRINT:INPUT"Quel jeu voulez-vou
s traiter ";na$:PRINT:INPUT"Version No "
;ve$:na$=UPPER$(na$)
50 PRINT:INPUT"Copyright ";cr$:PRINT:PRI
NT m3$
60 GOSUB 6000
200 CLS:PRINT m1$:PRINT m3$
210 PRINT TAB(10)"0 - Donnees"
220 PRINT TAB(10)"1 - Entrer lieux"
230 PRINT TAB(10)"2 - Entrer objets"
240 PRINT TAB(10)"3 - Entrer verbes"
250 PRINT TAB(10)"4 - positions depart"
260 PRINT TAB(10)"5 - connecter lieux"
270 PRINT TAB(10)"6 - entrer messages"
280 PRINT TAB(10)"7 - conditions et acti
ons"
290 PRINT TAB(10)"8 - enregistreur"
300 PRINT TAB(10)"9 - Fin du programme"
310 PRINT m3$
320 PRINT TAB(10)"Choisissez"
330 GOSUB 10000
340 IF a=9 THEN en=-1:GOTO 5100
350 ON a+1 GOTO 3000,1100,1200,1300,1400
,1500,4000,1600,5000,6000
499 REM ****** SOUS-PROGRAMME D'ENTREE/S
ORTIE
500 CLS
510 LOCATE 1,1:PRINT m4$:LOCATE 1,1:PRIN
T t1$
520 z=z+1
```

```
530 LOCATE 25,1:PRINT t2$;z
540 PRINT m3$
590 LOCATE 1,25:PRINT t3$;:eingabe$="":L
INE INPUT eingabe$
600 IF LEN(eingabe$)=0 THEN GOTO 200
610 IF a=2 THEN PRINT:LOCATE 1,25:INPUT"
Nom ";rn$(z):rn$(z)=UPPER$(rn$(z)):PRINT
620 PRINT
640 RETURN
1100 CLS:t1$="Entrer lieux ":t2$=" Lieu
No:":t3$="Je suis "
1105 z=ar
1110 GOSUB 510:       REM imprimer lieux
1120 ra$(z)=eingabe$
1125 ar=z
1130 GOTO 1110
1199 REM ************ FIN ENTREE LIEUX
1200 CLS:t1$="Entrer objets ":t2$=" Obje
t No:":t3$="Je vois "
1205 z=ao
1210 GOSUB 510
1220 ob$(z)=eingabe$
1225 ao=z
1230 GOTO 1210
1290 REM ********** FIN ENTREE OBJETS
1300 CLS:t1$="Entrer verbes":t2$="Verbe
No:":t3$=""
1305 z=av
1310 GOSUB 510
1320 ve$(z)=UPPER$(eingabe$)
1330 av=z
1360 GOTO 1310
1390 REM ********** FIN ENTREE VERBES
1400 CLS
1410 lp=13
```

```
1420 FOR i=lp TO ao
1430 GOSUB 12000:REM imprimer liste
1440 LOCATE 1,23:PRINT"Objet ";rn$(i);"
dans lieu No";:INPUT ob(i)
1450 lp=i+1
1460 CLS
1470 NEXT i
1480 GOTO 200
1500 FOR r1=i2 TO ar
1510 CLS
1515 PRINT
1520 GOSUB 12000
1530 PRINT"Lieu";r1;"mene au Nord a";:IN
PUT du(r1,1)
1540 PRINT"Lieu";r1;"mene au Sud a";:INP
UT du(r1,2)
1550 PRINT"Lieu";r1;"mene a l'Ouest a";:
INPUT du(r1,3)
1560 PRINT"Lieu";r1;"mene a l'Est a";:IN
PUT du(r1,4)
1570 PRINT"Lieu";r1;"mene en haut a";:IN
PUT du(r1,5)
1580 PRINT"Lieu";r1;"mene en bas a";:INP
UT du(r1,6)
1590 NEXT r1:GOTO 200
1600 CLS
1605 nb=0
1607 nb=nb+1
1608 IF nb>av THEN GOTO 200
1609 rn$(0)="<suite>"
1610 CLS:FOR i=0 TO ao
1620 PRINT i;rn$(i);"\";
1630 IF POS(#0)+LEN(rn$(i+1))>38 THEN PR
INT
1640 NEXT i
1650 PRINT:PRINT m3$:PRINT CHR$(10)
1700 PRINT"Action pour verbe ";ve$(nb);"
 et objet No";
1710 INPUT ob
1715 IF ob=0 THEN GOTO 1607
```

```
1800 CLS
1810 PRINT"Entrez conditions de l'action
 ":PRINT ve$(nb);" ";rn$(ob)
1815 PRINT m3$
1820 PRINT:PRINT:PRINT TAB(5)" R  - Obje
t est dans le lieu":PRINT
1830 PRINT TAB(5)" I  - Objet est dans l
'inventaire":PRINT
1840 PRINT TAB(5)" N  - Objet n'est pas
dans le lieu":PRINT
1850 PRINT TAB(5)" Fx - Flag est mis":PR
INT
1860 PRINT:PRINT TAB(5)" Gx - Flag est a
nnule":PRINT
1870 PRINT TAB(5)" Sxx- Joueur est dans
lieu xx"
1880 LOCATE 1,20:PRINT m3$;:PRINT"Ancien
 code ===> ";bc$(nb,ob):PRINT
1900 INPUT"Condition ";bc$(nb,ob):bc$(nb
,ob)=UPPER$(bc$(nb,ob))
1905 CLS:PRINT ve$(nb);" ";ob$(ob);" ent
raine, si:"
1906 PRINT bc$(nb,ob);" remplie, l'actio
n suivante:":PRINT m3$;
1910 PRINT TAB(5)"V  - ";rn$(ob);" dispa
rait":PRINT
1920 PRINT TAB(5)"I  - ";rn$(ob);" va da
ns INV":PRINT
1930 PRINT TAB(5)"Nxx- Objet xx reappara
it":PRINT
1940 PRINT TAB(5)"Dxy- Passage vers lieu
 x":PRINT
1950 PRINT TAB(5)"Sxx- Joueur vers lieu
xx":PRINT
1960 PRINT TAB(5)"Fx - Fixer flag x":PRI
NT
1970 PRINT TAB(5)"Lx - Annuler flag x":P
RINT
1980 PRINT TAB(5)"Mxx- Sortir message xx
":PRINT
1990 PRINT TAB(5)"T  - Joueur meurt":PRI
NT
```

```
2000 PRINT TAB(5)"E  - Fin car victoire"
2005 PRINT m3$;
2010 PRINT"Ancien code ===> ";ac$(nb,ob)
2020 INPUT"Action ";ac$(nb,ob):ac$(nb,ob
)=UPPER$(ac$(nb,ob))
2030 GOTO 1610
2040 REM ****** FIN ACTIONS ET CONDITION
S
2080 GOSUB 10000
2999 REM ************** CARACTERISTIQUES
3000 CLS:PRINT"Caracteristiques du jeu "
;na$
3010 PRINT m3$:PRINT
3020 PRINT" lieux: ";ar
3030 PRINT" Objets:";ao
3040 PRINT" Verbes: ";av
3050 PRINT" Condit: ";nb
3060 PRINT" Messag.:";am
3070 IF lo THEN PRINT" ";af;"Flags utili
ses"
3080 PRINT m3$
3090 PRINT"Debut jeu dans lieu:
";sp
3100 PRINT:PRINT"nombre de lettres a ent
rer: ";wl
3110 GOSUB 10000
3120 GOTO 200
3999 REM ****************** Messages
4000 CLS:PRINT"Entrer messages"
4010 PRINT m3$
4020 am=am+1
4030 PRINT am;:INPUT m$(am)
4040 IF LEN(m$(am))=0 THEN am=am-1:GOTO
200
4050 GOTO 4020
```

```
4060 REM ************* Fin MESSAGES
4999 REM ENREGISTREUR: sauver/charger
5000 CLS:PRINT"1 - Copie securite"
5010 PRINT:PRINT"2 - Charger fichier pou
r traitement"
5020 GOSUB 10000
5030 IF a<1 OR a>2 THEN GOTO 5020
5040 ON a GOTO 5100,5500
5100 CLS:PRINT"Attendre.":PRINT:PRINT"St
ockage en cours ..."
5110 OPENOUT na$
5120 PRINT#9,na$:PRINT#9,ve$:PRINT#9,cr$
:PRINT#9,wl
5130 PRINT#9,ar:PRINT#9,ao:PRINT#9,av:PR
INT#9,am:PRINT#9,sp:PRINT#9,af
5140 FOR i=1 TO ar:PRINT#9,ra$(i):NEXT i
5150 FOR i=1 TO ao:PRINT#9,ob$(i):PRINT#
9,rn$(i):PRINT#9,ob(i):NEXT i
5160 FOR i=1 TO av:PRINT#9,ve$(i):NEXT i
5170 FOR i=1 TO am:PRINT#9,m$(i):NEXT i
5180 FOR x=1 TO ar
5190 FOR y=1 TO 6
5200 PRINT#9,du(x,y)
5210 NEXT y
5220 NEXT x
5230 FOR x=1 TO av
5240 FOR y=1 TO ao
5250 PRINT#9,bc$(x,y)
5260 NEXT y
5270 NEXT x
```

```
5280 FOR x=1 TO av
5290 FOR y=1 TO ao
5300 PRINT#9,ac$(x,y)
5310 NEXT y
5320 NEXT x
5330 CLOSEOUT:IF en THEN CLS:END
5340 GOTO 200
5500 CLS:PRINT"Attendre.":PRINT:PRINT"Do
nnees en cours de chargement ..."
5510 OPENIN na$
5520 INPUT#9,na$:INPUT#9,ve$:INPUT#9,cr$
:INPUT#9,wl
5530 INPUT#9,ar:INPUT#9,ao:INPUT#9,av:IN
PUT#9,am:INPUT#9,sp:INPUT#9,af
5540 FOR i=1 TO ar:INPUT#9,ra$(i):NEXT i
5550 FOR i=1 TO ao:INPUT#9,ob$(i):INPUT#
9,rn$(i):INPUT#9,ob(i):NEXT i
5560 FOR i=1 TO av:INPUT#9,ve$(i):NEXT i
5570 FOR i=1 TO am:INPUT#9,m$(i):NEXT i
5580 FOR x=1 TO ar
5590 FOR y=1 TO 6
5600 INPUT#9,du(x,y)
5610 NEXT y
5620 NEXT x
5630 FOR x=1 TO av
5640 FOR y=1 TO ao
5650 INPUT#9,bc$(x,y)
5660 NEXT y
5670 NEXT x
5680 FOR x=1 TO av
5690 FOR y=1 TO ao
5700 INPUT#9,ac$(x,y)
5710 NEXT y
5720 NEXT x
```

```
5730 CLOSEIN
5740 GOTO 200
5999 REM ************ autres donnees
6000 CLS:PRINT"Entrez les autres donnees
 de controle   necessaires:":PRINT m3$
6010 PRINT:PRINT:INPUT"Longueur mot";wl
6020 PRINT:INPUT"Lieu de depart";sp
6030 PRINT:INPUT"Nombre flags";af
6040 RETURN
10000 a$=INKEY$:IF a$="" THEN 10000
10010 a=VAL(a$):RETURN
12000 LOCATE 1,1
12010 PRINT"Liste des lieux possibles:":
PRINT m3$
12020 FOR i1=1 TO ar
12030 PRINT i1;ra$(i1);
12040 IF POS(#0)+LEN(ra$(i1+1)) >38 THEN
 PRINT
12050 NEXT i1
12060 PRINT
12070 RETURN
```

INTERPRETEUR DE JEUX D'AVENTURE

```
1 REM ********************************
2 REM * Adventure System 1.0          *
3 REM * Adventureinterpreter          *
4 REM * (c) 1983,1984 by J. Walkowiak *
5 REM ********************************
6 CLS:PRINT"          Adventure System
1.0":PRINT:PRINT"           by Joerg Wal
kowiak":PRINT:PRINT:PRINT:PRINT"
 (c) 1984 by DATA BECKER":DEFINT a-z
10 FOR i=1 TO 5000:NEXT
40 CLEAR:CLS:INPUT"A quelle aventure vou
lez-vous jouer     ";na$
50 start=-1:GOSUB 900
60 INPUT#9,na$:CLS:PRINT na$;:INPUT#9,ve
$:PRINT"Version ";ve$:INPUT#9,cr$:INPUT#
9,wl:PRINT cr$
70 INPUT#9,ar:INPUT#9,ao:INPUT#9,av:INPU
T#9,am:INPUT#9,joueur:INPUT#9,af
75 DIM lieu$(ar),ob$(ao),rn$(ao),ob(ao),
verb$(av),m$(am),fl(af),ac$(av,ao),bc$(a
v,ao),ad$(20),passage(ar,6)
80 FOR i=1 TO ar:INPUT#9,lieu$(i):NEXT i
90 FOR i=1 TO ao:INPUT#9,ob$(i):INPUT#9,
nom$(i):INPUT#9,ob(i):nom$(i)=LEFT$(nom$
(i),wl):NEXT i
100 FOR i=1 TO av:INPUT#9,verb$(i):verb$
(i)=LEFT$(verb$(i),wl):NEXT i
110 FOR i=1 TO am:INPUT#9,m$(i):NEXT i
120 FOR x=1 TO ar:FOR y=1 TO 6
130 INPUT#9,passage(x,y)
140 NEXT y:NEXT x
150 FOR x=1 TO av:FOR y=1 TO ao
160 INPUT#9,bc$(x,y)
170 NEXT y:NEXT x
180 FOR x=1 TO av:FOR y=1 TO ao
190 INPUT#9,ac$(x,y)
200 NEXT y:NEXT x
210 CLOSEIN
880 GOTO 1000
```

```
900 MODE 1:PAPER 0:INK 0,0:BORDER 0:PEN
1:INK 1,25
910 PRINT"      CPC - Ventures"
920 PRINT TAB(15)CHR$(164);" 1984  by J"
;CHR$(178);"rg Walkowiak"
925 PRINT STRING$(40,208):PRINT"Imaginez
 un robot que vous pourriez     commande
r avec de nombreux ordres."
930 PRINT"Je suis ce robot et je vais m'
exposer   pour vous aux aventures les pl
us        risquees."
940 PRINT"Pour que vous puissiez me fair
e agir    utilement, je vous decrirai ch
aque fois precisement la situation dans
laquelle  je me trouve."
950 PRINT"Vous me direz ensuite avec deu
x mots    comme par exemple PRENDS COUTE
AU ce que je dois faire.":PRINT
960 PRINT"Je comprends en outre les inst
ructions  INVENTAIRE, INSTRUCTIONS et EN
D.":IF NOT start THEN GOTO 980
970 start=0:PRINT STRING$(40,210) :PRINT
"Veuillez patienter - Je charge le jeu !
":pg$="!"+na$:OPENIN pg$
980 INK 3,12,24:INK 2,24,12:PEN 3: PRINT
"        Appuyez";:PEN 2:PRINT" sur une
<touche>";:PEN 1:PRINT" ..."
990 entree$=INKEY$:IF entree$="" THEN 99
0 ELSE CLS:MODE 1:INK 1,2:INK 2,14:INK 3
,26:RETURN
1000 MODE 1:PAPER 0:INK 0,0:BORDER 0:PEN
 1:INK 1,2:PEN 2:INK 2,14:PEN 3:INK 3,26

1010 lignevide$=STRING$(40,32)
1020 DATA le Nord, le Sud, l'Ouest, l'Es
t, en haut, en bas
1030 FOR direction=1 TO 6
1040 READ direction$(direction)
1050 NEXT direction
1070 CLS
1080 PRINT:PRINT
1090 LOCATE 1,1
```

```
1100 FOR ligne=1 TO 10
1110 PRINT lignevide$;
1120 NEXT ligne
1130 LOCATE 1,1:PEN 1
1140 PRINT"Je suis ";
1150 PRINT lieu$(joueur);:PRINT"."
1160 PRINT"Je vois ";:imprime=0
1170 FOR i=1 TO ao
1180 IF ob(i)<>joueur THEN 1210
1190 IF POS(#0)+LEN(ob$(i))+2<=39 THEN P
RINT ob$(i);", ";:GOTO 1205
1200 IF POS(#0)+LEN(ob$(i))+2>39 THEN PR
INT:GOTO 1190
1205 imprime=-1
1210 NEXT i
1215 IF NOT imprime THEN PRINT"rien de s
pecial  ";
1220 PRINT CHR$(8);CHR$(8);"."
1230 PRINT lignevide$:PEN 2
1240 PRINT"Je peux aller vers ";:imprime
=0
1250 FOR direction=1 TO 6
1260 IF passage(joueur,direction)=0 THEN
 GOTO 1300 ELSE imprime=-1
1270 IF POS(#0)=20 THEN PRINT direction$
(direction);:GOTO 1300
1280 IF POS(#0)+LEN(direction$(direction
))<38 THEN PRINT", ";direction$(directio
n);:GOTO 1300
1290 PRINT",":PRINT direction$(direction
);:GOTO 1300
1300 NEXT direction
1310 IF imprime=0 THEN PRINT"nulle part"
;
1320 PRINT".":PEN 3
1330 PRINT STRING$(40,154);
1390 PEN 3:LOCATE 1,25:INPUT"Que dois-je
 faire";entree$:entree$=UPPER$(entree$):
PEN 2
```

```
1400 IF LEN(entree$)>2 THEN 1500
1410 IF entree$="N" AND passage(joueur,1
)<>0 THEN joueur=passage(joueur,1):PRINT
"O.k.":GOTO 1080
1420 IF entree$="S" AND passage(joueur,2
)<>0 THEN joueur=passage(joueur,2):PRINT
"O.k.":GOTO 1080
1430 IF entree$="O" AND passage(joueur,3
)<>0 THEN joueur=passage(joueur,3):PRINT
"O.k.":GOTO 1080
1440 IF entree$="E" AND passage(joueur,4
)<>0 THEN joueur=passage(joueur,4):PRINT
"O.k.":GOTO 1080
1450 IF entree$="H" AND passage(joueur,5
)<>0 THEN joueur=passage(joueur,5):PRINT
"O.k.":GOTO 1080
1460 IF entree$="B" AND passage(joueur,6
)<>0 THEN joueur=passage(joueur,6):PRINT
"O.k.":GOTO 1080
1470 IF LEN(entree$)<3 THEN PRINT"Aucun
chemin n'y mene !":GOTO 1080
1480 IF LEN(entree$)>8 THEN GOTO 2000
1499 REM ********** DEBUT INVENTAI
RE
1500 IF LEFT$(entree$,3)<>"INV" THEN GOT
O 1600
1510 PRINT"Voici ce que je porte sur moi
:"
1520 FOR objet=1 TO ao
1530 IF ob(objet)=-1 THEN PRINT ob$(obje
t)
1540 NEXT objet
1550 GOTO 1080
1560 REM ********** FIN INVENTAIRE
1599 REM ********** SAVE GAME
1600 IF LEFT$(entree$,3)<>"SAV" THEN GOT
O 1700
1620 OPENOUT "etatjeu"
1625 PRINT#9,joueur
1630 FOR objet=1 TO ao
```

```
1631 PRINT#9,ob(objet)
1632 NEXT objet
1635 FOR lieu=1 TO ar
1636 FOR direction=1 TO 6
1637 PRINT#9,passage(lieu,direction)
1638 NEXT direction
1639 NEXT lieu
1645 FOR flag=1 TO af
1646 PRINT#9,fl(flag)
1647 NEXT flag
1650 CLOSEOUT
1670 GOTO 1080
1699 REM ********************* LOAD GAME
1700 IF LEFT$(entree$,3)<>"LOA" THEN GOT
O 1900
1720 OPENIN "etatjeu"
1725 INPUT#9,joueur
1730 FOR objet=1 TO ao
1731 INPUT#9,ob(objet)
1732 NEXT objet
1735 FOR lieu=1 TO ar
1736 FOR direction=1 TO 6
1737 INPUT#9,passage(lieu,direction)
1738 NEXT direction
1739 NEXT lieu
1745 FOR flag=1 TO af
1746 INPUT#9,fl(flag)
1747 NEXT flag
1750 CLOSEIN
1770 GOTO 1080
1900 IF LEFT$(entree$,3)<>"INS" THEN GOT
O 1950
1910 GOSUB 900
1920 GOTO 1080
1950 IF LEFT$(entree$,3)<>"END" THEN GOT
O 2000 ELSE CLS
1960 PRINT"Nous vous souhaitons plus de
succes la  prochaine fois !":PRINT:PRINT
:PRINT:END
```

```
2000 longueur=LEN(entree$)
2010 FOR lettre=1 TO longueur
2020 tester$=MID$(entree$,lettre,1)
2030 IF tester$<>" "THEN NEXT lettre
2040 everb$=LEFT$(entree$,wl)
2050 rl=longueur-lettre
2060 IF rl<0 THEN 2090
2070 eobjet$=RIGHT$(entree$,rl)
2080 eobjet$=LEFT$(eobjet$,wl)
2090 FOR vn=1 TO av
2100 IF everb$=verb$(vn) THEN 2130
2110 NEXT vn
2120 PRINT"Je ne comprends pas ce verbe
!":GOTO 1080
2130 FOR o=1 TO ao
2140 IF eobjet$=nom$(o) THEN 2200
2150 NEXT o
2160 PRINT"Je ne connais pas cet objet !
":GOTO 1080
2198 REM vn & o sont les numeros de verb
e et d'objet
2199 REM ********** CONDITIONS REMPLIES
2200 FOR ab=1 TO LEN(bc$(vn,o))
2210 bd$(ab)=MID$(bc$(vn,o),ab,1)
2220 NEXT ab
2250 FOR aa=1 TO LEN(ac$(vn,o))
2260 ad$(aa)=MID$(ac$(vn,o),aa,1)
2270 NEXT aa
2280 REM conditions en bd$
         actions en ad$
2290 ab=ab-1:aa=aa-1
2300 x=0:er=-1:ok=-1
2310 x=x+1:ok=(ok AND er):er=0
2320 IF x=ab+1 THEN 2500
2330 IF bd$(x)<>"R" THEN 2350
2340 IF ob(o)=joueur THEN er=-1
2345 GOTO 2310
2350 IF bd$(x)<>"I" THEN 2370
2360 IF ob(o)=-1 THEN er=-1
2365 GOTO 2310
2370 IF bd$(x)<>"N" THEN 2390
```

```
2380 IF (ob(o)<>joueur AND ob(o))<>-1) T
HEN er=-1
2385 GOTO 2310
2390 IF bd$(x)<>"S" THEN 2410
2400 r1$=bd$(x+1)+bd$(x+2):x=x+2:IF joue
ur=VAL(r1$) THEN er=-1
2405 GOTO 2310
2410 IF bd$(x)<>"F" THEN 2450
2420 IF fl(VAL(bd$(x+1)))=-1 THEN 2440
2430 x=x+1:GOTO 2450
2440 er=-1:x=x+1:GOTO 2310
2450 IF bd$(x)<>"G" THEN 2310
2460 IF NOT fl(VAL(bd$(x+1))) THEN 2480
2470 x=x+1:GOTO 2310
2480 er=-1:x=x+1:GOTO 2310
2490 GOTO 2310
2500 IF NOT ok THEN PRINT"Cela ne marche
 pas pour le moment !":GOTO 1080
3999 REM **** TOUTES CONDITIONS REMPLIES
4000 PRINT"Okay !"
4999 REM ********** EXECUTER ACTIONS
5000 x=0:er=0
5040 x=x+1
5050 IF x>AA THEN 1080
5060 IF ad$(x)<>"V" THEN 5080
5070 ob(o)=0:GOTO 5040
5080 IF ad$(x)<>"I" THEN 5100
5090 ob(o)=-1:GOTO 5040
5100 IF ad$(x)<>"N" THEN 5120
5110 GOSUB 5500:ob(pa)=joueur:ac$(vn,o)=
"M00":GOTO 5040
5120 IF ad$(x)<>"F" THEN 5140
5130 GOSUB 5600:fl(pa)=-1:GOTO 5040
5140 IF ad$(x)<>"L" THEN 5160
5150 GOSUB 5600:fl(pa)=0:GOTO 5040
5160 IF ad$(x)<>"M" THEN 5180
5170 GOSUB 5500:PRINT m$(pa):GOTO 5040
```

```
5180 IF ad$(x)<>"E" THEN 5200
5190 GOTO 6000
5200 IF ad$(x)<>"D" THEN 5040
5210 GOSUB 5700
5220 passage(joueur,p1)=p2
5230 IF p1=1 THEN p3=2 ELSE IF p1=2 THEN
 p3=1 ELSE IF p1=3 THEN p3=4 ELSE IF p1=
4 THEN p3=3 ELSE IF p1=5 THEN p3=6 ELSE
IF p1=6 THEN p3=5
5240 du(p2,p3)=joueur:GOTO 5040
5500 r1$=ad$(x+1)+ad$(x+2):x=x+2:pa=VAL(
r1$):RETURN
5600 r1$=ad$(x+1):x=x+1:pa=VAL(r1$):RETU
RN
5700 p1=VAL(ad$(x+1)):p2=VAL(ad$(x+2)):x
=x+2:RETURN
6000 CLS:PRINT"Toutes nos felicitations
!":PRINT:PRINT
6010 PRINT"Tu as reussi !":PRINT
6020 PRINT STRING$(17,10):END
```

Ce programme vous confrontera à toutes les questions auxquelles on doit trouver une réponse pour construire un programme de jeu d'aventure complet et fonctionnel.

Avant le lancement de 'Venturefix' l'action doit être déjà fixée, vous devez avoir établi une liste de tous les lieux, objets, mots et informations et vous devriez également avoir formulé déjà toutes les actions de façon textuelle.

Le travail créatif ne vous sera donc pas retiré, mais seulement le travail de routine de programmation.
Vous pouvez ignorer ce qu'est une instruction PRINT, pourtant vous pouvez construire des programmes BASIC irréprochables.

Pendant le dialogue Venturefix exigera de vous toutes les données nécessaires, et ce dans l'ordre, et vous informera en permanence du développement, jusqu'à ce qu'enfin un jeu d'aventure prêt ait été créé, dans le style des programmes présentés dans cet ouvrage.

```
10 REM ********************************
20 REM * aventuregenerator VENTUREFIX *
30 REM * (c) 1984   by Joerg Walkowiak *
40 REM ********************************

50 MODE 1:DEFINT a-z
60 PRINT"Venturefix":PRINT:PRINT"written
 by Joerg Walkowiak":LOCATE 1,24:PRINT"V
enturefix":PRINT"(c) 1984  by DATA BECKE
R, Duesseldorf";
70 FOR i=1 TO 10000:NEXT
80 WINDOW#0,1,40,1,20
90 WINDOW#1,1,40,19,25
100 GOTO 170
110 IF impression THEN PRINT#8,numerolig
ne;" ";z$
120 z1$=STR$(numeroligne)+" "+z$:PRINT#1
,z1$
130 PRINT#9,numeroligne;" ";z$:numerolig
ne=numeroligne+espacementlignes:z$="":RE
TURN
140 z$=z$+h$:RETURN
150 entree$=INKEY$:IF entree$="" THEN 15
0 ELSE RETURN
160 CLS:PLOT 0,115:DRAW 640,115:PRINT t$
:PRINT STRING$(40,154):RETURN
170 CLS:REM * debut programme principal
180 INPUT"Comment doit s'appeler l'avent
ure";entree$:entree$=UPPER$(entree$)
190 IF LEN(entree$)>8 THEN PRINT"Bitte n
icht mehr als 8 lettren !":GOTO 180
200 aventure$=entree$+".BAS"
210 PRINT:INPUT"Faut-il sortir en meme t
emps un listing sur une imprimante conne
ctee ";entree$
220 IF (entree$="o" OR entree$="O") THEN
 impression=-1
230 CLS:INPUT"Nom de l'auteur";auteur$
240 PRINT aventure$;:INPUT", Version ";v
ersion$
```

```
250 INPUT"Date ";date$
260 PRINT:PRINT:LINE INPUT"Placez une di
squette/cassette  vide et  appuyez sur <
ENTER> ...",entree$
270 t$="Entrer donnees de controle"
280 CLS#1:GOSUB 160
290 INPUT"Combien de lieux      ";ar
300 INPUT"Combien d'objets      ";ao
310 INPUT"Combien de verbes     ";av
320 INPUT"Evaluation: Flags     ";af
330 INPUT"Lieu de debut du jeu ";joueur
340 PRINT:INPUT"Nombre de lettres signif
icatives";wl
350 t$="PARTIE I : Initialisation":GOSUB
 160
360 PRINT"- ouvrir fichier":aventure$="!
"+ UPPER$(aventure$):OPENOUT aventure$
370 PRINT"- Tete de programme":numerolig
ne=1:espacementlignes=1:z$="REM "+LEFT$(
aventure$,(LEN(aventure$)-4))+", Version
 "+version$:GOSUB 110
380 z$="REM (c) "+date$:GOSUB 110
390 z$="REM by  "+auteur$:GOSUB 110
400 z$="REM produit avec l'aide de:":GOS
UB 110
410 z$="REM VENTUREFIX, (c) 1984 by J. W
alkowiak":GOSUB 110
420 z$="REM"+STRING$(35,154):GOSUB 110
430 PRINT"- Titre":z$="REM programmer ic
i page titre":numeroligne=10
440 PRINT"- Caracteristiques":numerolign
e=150:z$="ar="+STR$(ar)+":ao="+STR$(ao)+
":av="+STR$(av)+":af="+STR$(af):espaceme
ntlignes=10:GOSUB 110
450 z$="joueur="+STR$(joueur):GOSUB 110
460 z$="wl="+STR$(wl):GOSUB 110
470 PRINT"- dimensionner les tableaux"
480 z$="DIM lieu$(ar),passage(ar,6),ob$(
ao),nom$(ao),ob(ao),verb$(av)":numerolig
ne=190:GOSUB 110
```

```
490 z$="REM ************************ VE
RBES":GOSUB 110:espacementlignes=1
500 t$="Entrez maintenant chaque fois un
 verbe  et appuyez sur <ENTER>:":GOSUB 1
60
510 FOR i=1 TO av
520 PRINT"Verbe ";i;:INPUT"est ";verb$
530 z$="DATA "+verb$
540 GOSUB 110
550 NEXT i
560 numeroligne=300:z$="REM ***********
******* OBJETS":GOSUB 110
570 CLS:t$="Entrez maintenant chaque foi
s un objet  et appuyez sur <ENTER>:":GOS
UB 160
580 FOR i=1 TO ao
590 CLS:GOSUB 160
600 LOCATE 1,5:PRINT"Objet No:   ";i
610 INPUT"Je vois ";objet$
620 INPUT"Nom     ";nom$
630 PRINT
640 INPUT"dans le lieu No";lieu$
650 z$="DATA "+objet$+","+nom$+","+lieu$
:GOSUB 110
660 NEXT i
670 t$="Entrez maintenant la description
 de     chaque lieu ainsi que les numero
s des   lieux auxquels on peut acceder a
 partir de ce lieu.":GOSUB 160
680 numeroligne=500
690 z$="REM ************ DESCRIPTIONS D
E LIEUX":GOSUB 110
700 FOR i=1 TO ar
710 CLS:GOSUB 160
720 LOCATE 1,5:PRINT"Lieu No:   ";i
730 INPUT"Je suis ";lieu$
740 PRINT:PRINT"Ce lieu mene"
```

```
750 INPUT" au Nord au lieu No     ";nord$
760 INPUT" au Sud au lieu No      ";sud$
770 INPUT" a l'Ouest au lieu No   ";ouest
$
780 INPUT" a l'Est au lieu No     ";est$
790 INPUT" en haut au lieu No     ";haut$
800 INPUT" en bas au lieu No      ";bas$
810 z$="DATA "+lieu$+","+nord$+","+sud$+
","+ouest$+","+est$+","+haut$+","+bas$:G
OSUB 110
820 NEXT i
830 t$="Vous disposez des messages stand
ard     suivants":GOSUB 160
840 PRINT"m$(1): Je ne vois rien de spec
ial."
850 PRINT"m$(2): Je ne suis pas si fort
!"
860 PRINT"m$(3): Comment vois-tu cela ?"
870 PRINT"m$(4): Je ne comprends pas ce
que tu veux      dire !"
880 PRINT:PRINT"ok$:   D'accord !"
890 numeroligne=600:z$="REM ************
******* MESSAGES":GOSUB 110
900 z$="m$(1)="+CHR$(34)+"Je ne vois rie
n de special."+CHR$(34):GOSUB 110
910 z$="m$(2)="+CHR$(34)+"Je ne suis pas
 si fort !"+CHR$(34):GOSUB 110
920 z$="m$(3)="+CHR$(34)+"Comment vois-t
u cela ?"+CHR$(34):GOSUB 110
930 z$="m$(4)="+CHR$(34)+"Je ne comprend
s pas ce que tu veux dire !"+CHR$(34):GO
SUB 110
940 z$="ok$="+CHR$(34)+"D'accord !"+CHR$
(34):GOSUB 110
950 t$="Partie II":GOSUB 160
960 numeroligne=699:z$="REM ***********
ici eventuellement second titre":GOSUB 1
10
```

```
970 PRINT"- ecrire partie d'initialisati
on"
980 numeroligne=800:espacementlignes=10
990 z$="FOR i=1 TO av":GOSUB 110
1000 z$="READ verb$(i):verb$(i)=LEFT$(ve
rb$(i),wl):verb$(i)=UPPER$(verb$(i))":GO
SUB 110
1010 z$="NEXT i":GOSUB 110
1020 z$="FOR objet=1 TO ao":GOSUB 110
1030 z$="READ ob$(objet),nom$(objet),ob(
objet):nom$(objet)=LEFT$(nom$(objet),wl)
:nom$(objet)=UPPER$(nom$(objet)):NEXT ob
jet":GOSUB 110
1040 z$="FOR lieu=1 TO ar:READ lieu$(lie
u)":GOSUB 110
1050 z$="FOR direction=1 TO 6:READ passa
ge(lieu,direction)":GOSUB 110
1060 z$="NEXT direction:NEXT lieu":GOSUB
 110
1070 PRINT"- Explication du jeu"
1080 z$="PRINT:INPUT"+CHR$(34)+"   Voule
z-vous des conseils"+CHR$(34)+";entree$:
entree$=UPPER$(entree$)":GOSUB 110
1090 z$="IF LEFT$(entree$,1)="+CHR$(34)+
"O"+CHR$(34)+"THEN GOSUB 900:GOTO 1000
ELSE GOTO 1000":GOSUB 110
1100 z$="MODE 1:PAPER 0:INK 0,0:BORDER 0
:PEN 1:INK 1,25":GOSUB 110
1110 Z$="PRINT"+CHR$(34)+"      CPC - Ve
ntures"+CHR$(34)+":PRINT TAB(15)CHR$(164
);"+CHR$(34)+" 1984  by J"+CHR$(34)+"CHR
$(178);"+CHR$(34)+"rg Walkowiak":GOSUB 1
10
1120 z$="PRINT STRING$(40,208):PRINT"+CH
R$(34)+"Imaginez un robot que vous pourr
iez     commander avec de nombreux ordre
s."+CHR$(34):GOSUB 110
```

```
1130 z$="PRINT"+CHR$(34)+"Je suis ce rob
ot et je vais m'exposer   pour vous aux
aventures les plus        risquees."+CHR
$(34):GOSUB 110
1140 z$="PRINT"+CHR$(34)+"Pour que vous
puissiez me faire agir    utilement, je
vous decrirai chaque fois precisement la
 situation dans laquelle  je me trouve."
+CHR$(34):GOSUB 110
1150 z$="PRINT"+CHR$(34)+"Vous me direz
ensuite avec deux mots    comme par exem
ple PRENDS COUTEAU ce que je dois faire.
"+CHR$(34)+":PRINT":GOSUB 110
1160 z$="PRINT"+CHR$(34)+"Je comprends e
n outre les instructions  INVENTAIRE, SA
VE, LOAD et END."+CHR$(34):GOSUB 110
1170 z$="PRINT STRING$(40,210):INK 3,12,
24:INK 2,24,12:PEN 3: PRINT"+CHR$(34)+"
       Appuyez sur"+CHR$(34)+";:PEN 2:PR
INT"+CHR$(34)+" <une touche>"+CHR$(34)+"
;:PEN 1:PRINT":GOSUB 110
1180 z$="entree$=INKEY$:IF entree$="+STR
ING$(2,34)+" THEN 980 ELSE CLS:MODE 1:IN
K 1,2:INK 2,14:INK 3,26:RETURN":GOSUB 11
0
1190 t$="Partie III: Aventure":GOSUB 160

1200 z$="MODE 1:PAPER 0:INK 0,0:BORDER 0
:INK 1,2:INK 2,14:INK 3,26:PEN 1":GOSUB
110
1210 z$="lignevide$=STRING$(40,32)":GOSU
B 110
1220 z$="DATA le Nord,le Sud,l'Ouest,l'E
st,en haut,en bas":GOSUB 110
1230 z$="FOR direction=1 to 6":GOSUB 110
1240 z$="READ direction$(direction)":GOS
UB 110
1250 z$="NEXT direction":GOSUB 110
1260 numeroligne=1070:z$="CLS":GOSUB 110
```

```
1270 z$="PRINT:PRINT":GOSUB 110
1280 PRINT"- Sortie sur ecran"
1290 z$="LOCATE 1,1":GOSUB 110
1300 z$="FOR ligne=1 TO 10":GOSUB 110
1310 z$="PRINT lignevide$;":GOSUB 110
1320 z$="NEXT ligne":GOSUB 110
1330 z$="LOCATE 1,1:PEN 1":GOSUB 110
1340 z$="PRINT"+CHR$(34)+"Je suis "+CHR$
(34)+";":GOSUB 110
1350 z$="PRINT lieu$(joueur);:PRINT"+CHR
$(34)+"."+CHR$(34):GOSUB 110
1360 z$="PRINT"+CHR$(34)+"Je vois "+CHR$
(34)+";:imprime=0":GOSUB 110
1370 z$="FOR i=1 TO ao":GOSUB 110
1380 z$="IF ob(i)<>joueur THEN 1210":GOS
UB 110
1390 z$="IF POS(#0)+LEN(ob$(i))+2<=39 TH
EN PRINT ob$(i);"+CHR$(34)+", "+CHR$(34)
+";:GOTO 1205":GOSUB 110
1400 z$="IF POS(#0)+LEN(ob$(i))+2>39 THE
N PRINT:GOTO 1190":GOSUB 110
1410 z$="imprime=-1":numeroligne=1205:GO
SUB 110:numeroligne=1210
1420 z$="NEXT i":GOSUB 110
1430 z$="IF NOT imprime THEN PRINT"+CHR$
(34)+"rien de special  "+CHR$(34)+";":nu
meroligne=1215:GOSUB 110:numeroligne=122
0
1440 z$="PRINT CHR$(8);CHR$(8);"+CHR$(34
)+"."+CHR$(34):GOSUB 110
1450 z$="PRINT lignevide$:PEN 2":GOSUB 1
10
1460 z$="PRINT"+CHR$(34)+"Je peux aller
vers "+CHR$(34)+";:imprime=0":GOSUB 110
1470 z$="FOR direction=1 TO 6":GOSUB 110
1480 z$="IF passage(joueur,direction)=0
THEN GOTO 1300 ELSE imprime=-1":GOSUB 11
0
```

```
1490 z$="IF POS(#0)=20 THEN PRINT direct
ion$(direction);:GOTO 1300":GOSUB 110
1500 z$="IF POS(#0)+LEN(direction$(direc
tion))<38 THEN PRINT"+CHR$(34)+", "+CHR$
(34)+";direction$(direction);:GOTO 1300"
:GOSUB 110
1510 z$="PRINT"+CHR$(34)+","+CHR$(34)+":
PRINT direction$(direction);:GOTO 1300":
GOSUB 110
1520 z$="NEXT direction":GOSUB 110
1530 z$="IF imprime=0 THEN PRINT"+CHR$(3
4)+"nulle part"+CHR$(34)+";":GOSUB 110
1540 z$="PRINT"+CHR$(34)+"."+CHR$(34)+":
PEN 3":GOSUB 110
1550 z$="PRINT STRING$(40,154)":GOSUB 11
0
1560 PRINT"- Entree du coup"
1570 numeroligne=1390:z$="PEN 3:LOCATE 1
,25:INPUT"+CHR$(34)+"Que dois-je faire"+
CHR$(34)+";entree$:entree$=UPPER$(entree
$):PEN 2":GOSUB 110
1580 z$="IF LEN(entree$)>2 THEN 1500":GO
SUB 110
1590 z$="IF entree$="+CHR$(34)+"N"+CHR$(
34)+" AND passage(joueur,1)<>0 THEN joue
ur=passage(joueur,1):PRINT ok$:GOTO 1080
":GOSUB 110
1600 z$="IF entree$="+CHR$(34)+"S"+CHR$(
34)+" AND passage(joueur,2)<>0 THEN joue
ur=passage(joueur,2):PRINT ok$:GOTO 1080
":GOSUB 110
1610 z$="IF entree$="+CHR$(34)+"O"+CHR$(
34)+" AND passage(joueur,3)<>0 THEN joue
ur=passage(joueur,3):PRINT ok$:GOTO 1080
":GOSUB 110
1620 z$="IF entree$="+CHR$(34)+"E"+CHR$(
34)+" AND passage(joueur,4)<>0 THEN joue
ur=passage(joueur,4):PRINT ok$:GOTO 1080
":GOSUB 110
```

```
1630 z$="IF entree$="+CHR$(34)+"H"+CHR$(
34)+" AND passage(joueur,5)<>0 THEN joue
ur=passage(joueur,5):PRINT ok$:GOTO 1080
":GOSUB 110
1640 z$="IF entree$="+CHR$(34)+"B"+CHR$(
34)+" AND passage(joueur,6)<>0 THEN joue
ur=passage(joueur,6):PRINT ok$:GOTO 1080
":GOSUB 110
1650 z$="IF LEN(entree$)<3 THEN PRINT"+C
HR$(34)+"Aucun chemin n'y mene !"+CHR$(3
4)+":GOTO 1080":GOSUB 110
1660 z$="IF LEN(entree$)>8 THEN GOTO 200
0":GOSUB 110
1670 PRINT"- INVentaire"
1680 numeroligne=1499:z$="REM **********
****** DEBUT INVENTAIRE":GOSUB 110
1690 numeroligne=1500:z$="IF LEFT$(entre
e$,3)<>"+CHR$(34)+"INV"+CHR$(34)+" THEN
GOTO 1600":GOSUB 110
1700 z$="PRINT"+CHR$(34)+"Voici ce que j
e porte sur moi:"+CHR$(34):GOSUB 110
1710 z$="FOR objet=1 TO ao":GOSUB 110
1720 z$="IF ob(objet)=-1 THEN PRINT ob$(
objet)":GOSUB 110
1730 z$="NEXT objet":GOSUB 110
1740 z$="GOTO 1080":GOSUB 110
1750 z$="REM *************** FIN INVENT
AIRE":GOSUB 110
1760 PRINT"- SAVe Game":numeroligne=1600
1770 z$="IF LEFT$(entree$,3)<>"+CHR$(34)
+"SAV"+CHR$(34)+" THEN GOTO 1700":GOSUB
110
1780 z$="PRINT"+CHR$(34)+"Appuyez sur RE
C & PLAY ...                Sous quel nom
sauvegarder ............."+CHR$(34)+";S
TRING$(10,8);:INPUT entree$":GOSUB 110
```

```
1790 z$="IF LEN(entree$)>10 THEN PRINT"+
CHR$(34)+"Un peu plus court S.V.P. !"+CH
R$(34)+":GOTO 1610 ELSE entree$="+CHR$(3
4)+"!"+CHR$(34)+"+entree$+"+CHR$(34)+".j
eu"+CHR$(34)+":OPENOUT entree$":GOSUB 11
0
1800 z$="PRINT#9,joueur":GOSUB 110
1810 z$="FOR objet=1 TO ao":GOSUB 110
1820 z$="PRINT#9,ob(objet)":GOSUB 110
1830 z$="NEXT objet":GOSUB 110
1840 z$="FOR lieu=1 TO ar:FOR direction=
1 TO 6:PRINT#9,passage(lieu,direction):N
EXT direction:NEXT lieu":GOSUB 110
1850 z$="FOR flag=1 TO af:PRINT#9,fl(fla
g):NEXT flag":GOSUB 110
1860 z$="CLOSEOUT:PRINT ok$:GOTO 1080":G
OSUB 110
1870 PRINT"- LOAd Game"
1880 z$="IF LEFT$(entree$,3)<>"+CHR$(34)
+"LOA"+CHR$(34)+" THEN GOTO 1900":GOSUB
110
1890 z$="PRINT"+CHR$(34)+"Rembobiner cas
sette et appuyer sur PLAY Quel jeu faut-
il charger ..."+CHR$(34)+";STRING$(4,8);
:INPUT entree$":GOSUB 110
1900 z$="IF LEN(entree$)>10 THEN PRINT"+
CHR$(34)+"Ce n'est pas possible !"+CHR$(
34)+":GOTO 1710 ELSE entree$="+CHR$(34)+
"!"+CHR$(34)+"+entree$+"+CHR$(34)+".JEU"
+CHR$(34)+":OPENIN entree$":GOSUB 110
1910 z$="INPUT#9,joueur":GOSUB 110
1920 z$="FOR objet=1 TO ao:INPUT#9,ob(ob
jet):NEXT objet":GOSUB 110
1930 z$="FOR lieu=1 TO ar:FOR direction=
1 TO 6:INPUT#9,passage(lieu,direction):N
EXT direction:NEXT lieu":GOSUB 110
1940 z$="FOR flag=1 TO af:INPUT#9,fl(fla
g):NEXT flag":GOSUB 110
1950 z$="CLOSEIN:PRINT ok$:GOTO 1080":GO
SUB 110
```

```
1960 PRINT"- INS":numeroligne=1900
1970 z$="IF LEFT$(entree$,3)<>"+CHR$(34)
+"INS"+CHR$(34)+" THEN GOTO 1950":GOSUB
110
1980 z$="GOSUB 900:GOTO 1080":GOSUB 110
1990 PRINT"- END":numeroligne=1950
2000 z$="IF LEFT$(entree$,3)<>"+CHR$(34)
+"END"+CHR$(34)+" THEN GOTO 2000":GOSUB
110
2010 z$="PRINT"+CHR$(34)+"Nous vous souh
aitons plus de succes la  prochaine fois
."+CHR$(34)+":PRINT:PRINT:PRINT:END":GOS
UB 110
2020 PRINT"- Analyse de l'entree":numero
ligne=2000
2030 z$="longueur=LEN(entree$)":GOSUB 11
0
2040 z$="FOR lettre=1 TO longueur":GOSUB
 110
2050 z$="tester$=MID$(entree$,lettre,1)"
:GOSUB 110
2060 z$="IF tester$<>"+CHR$(34)+CHR$(32)
+CHR$(34)+"THEN NEXT lettre":GOSUB 110
2070 z$="everb$=LEFT$(entree$,wl)":GOSUB
 110
2080 z$="rl=longueur-lettre":GOSUB 110
2090 z$="IF rl<0 THEN 2090":GOSUB 110
2100 z$="eobjet$=RIGHT$(entree$,rl)":GOS
UB 110
2110 z$="eobjet$=LEFT$(eobjet$,wl)":GOSU
B 110
2120 z$="FOR numeroverb=1 TO av":GOSUB 1
10
2130 z$="IF everb$=verb$(numeroverb) THE
N 2130":GOSUB 110
2140 z$="NEXT numeroverb":GOSUB 110
2150 z$="PRINT"+CHR$(34)+"Je ne comprend
s pas ce verbe."+CHR$(34)+":GOTO 1080":G
OSUB 110
2160 z$="FOR o=1 TO ao":GOSUB 110
2170 z$="IF eobjet$=nom$(o) THEN 2200":G
OSUB 110
```

```
2180 z$="NEXT o":GOSUB 110
2190 z$="PRINT"+CHR$(34)+"Je ne connais
pas cet objet."+CHR$(34)+":GOTO 1080":GO
SUB 110
2200 PRINT"- Table de saut":numeroligne=
2200:butdusaut=5000
2210 z$="ON numeroverb GOTO 5000"
2220 FOR i=1 TO av
2230 butdusaut=butdusaut+1000
2240 z$=z$+","+STR$(butdusaut)
2250 NEXT i
2260 GOSUB 110
2270 PRINT"- Titre: joueur mort":numerol
igne=4500
2280 z$="CLS:REM JOUEUR MORT":GOSUB 110
2290 z$="PRINT"+CHR$(34)+"Rien ne m'aura
 ete epargne !"+CHR$(34)+":PRINT:PRINT m
$(0)":GOSUB 110
2300 z$="PRINT:PRINT"+CHR$(34)+"Je suis
mort !"+CHR$(34)+":PRINT":GOSUB 110
2310 z$="INPUT"+CHR$(34)+"Dois-je essaye
r encore une fois "+CHR$(34)+";entree$:e
ntree$=UPPER$(entree$)":GOSUB 110
2320 z$="IF LEFT$(entree$,1)="+CHR$(34)+
"O"+CHR$(34)+" THEN RUN":GOSUB 110
2330 z$="GOTO 1960":GOSUB 110
2340 PRINT"- Titre: JOUEUR GAGNE":numero
ligne=4800
2350 z$="CLS:REM JOUEUR GAGNE":GOSUB 110
2360 z$="PRINT"+CHR$(34)+"Toutes nos fel
icitations !"+CHR$(34):GOSUB 110
2370 z$="PRINT:PRINT"+CHR$(34)+"Vous ave
z accompli la mission qui vous  etait co
nfiee et vous pouvez maintenant vous lan
cer dans une autre aventure."+CHR$(34):G
OSUB 110
2380 z$="PRINT:PRINT:PRINT:PRINT:END":GO
SUB 110
2390 t$="Partie IV: Conditions et action
s":GOSUB 160
```

```
2400 espacementlignes=1:z1=5000:numeroli
gne=z1:o=0:v=0
2410 v=v+1
2420 o=o+1
2430 x=o:z$="IF o="+STR$(o)+" AND "
2440 GOSUB 160
2450 PRINT" 1 - l'objet est dans le lieu
"
2460 PRINT" 2 - l'objet n'est pas dans l
e lieu"
2470 PRINT" 3 - le joueur a l'objet"
2480 PRINT" 4 - le flag est mis"
2490 PRINT" 5 - le flag n'est pas mis"
2500 PRINT" 6 - le joueur doit etre dans
 lieu: ..."
2510 PRINT" 7 - ET autre condition"
2520 PRINT" 8 - OU autre condition"
2530 PRINT:PRINT" 0 - les conditions son
t Okay"
2540 PRINT
2550 PRINT"Choisissez pour verbe ";v;" o
bjet ";o
2560 INPUT e
2570 IF e=1 THEN z$=z$+"ob(o)=joueur"
2580 IF e=2 THEN z$=z$+"ob(o)<>joueur"
2590 IF e=3 THEN INPUT"objet numero ";x:
z$=z$+"ob("+STR$(x)+")=joueur"
2600 IF e=4 THEN INPUT"Flag numero   ";x
:z$=z$+"fl("+STR$(x)+")=-1"
2610 IF e=5 THEN INPUT"Flag numero   ";x
:z$=z$+"fl("+STR$(x)+")<>-1"
2620 IF e=6 THEN INPUT"Quel lieu     ";x
:z$=z$+"joueur="+STR$(x)
2630 IF e=7 THEN z$=z$+" AND "
2640 IF e=8 THEN z$=z$+" OR "
2650 IF e<>0 THEN 2440
2660 z$=z$+" THEN "
2670 GOSUB 160
2680 PRINT" 1 - objet disparait"
2690 PRINT" 2 - objet dans inventaire"
2700 PRINT" 3 - objet apparait encore"
2710 PRINT" 4 - Flag est mis"
```

```
2720 PRINT" 5 - Flag est supprime"
2730 PRINT" 6 - Ouvrir passage"
2740 PRINT" 7 - Sortir message"
2750 PRINT" 8 - Joueur meurt"
2760 PRINT" 9 - Joueur gagne"
2770 PRINT:PRINT" 0 - Actions Okay"
2780 PRINT"Choisissez pour verbe ";v;" o
bjet ";o
2790 IF e=1 THEN INPUT"objet numero ";x:
z$=z$+"ob("+STR$(x)+")=0"
2800 IF e=2 THEN INPUT"objet numero ";x:
z$=z$+"ob("+STR$(x)+")=-1"
2810 IF e=3 THEN INPUT"objet numero ";x:
z$=z$+"ob("+STR$(x)+")=joueur"
2820 IF e=4 THEN INPUT"Flag numero ";x:z
$=z$+"fl("+STR$(x)+")=-1"
2830 IF e=5 THEN INPUT"Flag numero ";x:z
$=z$+"fl("+STR$(x)+")=0"
2840 IF e=6 THEN INPUT"du lieu ";x:INPUT
" dans la direction ";y:INPUT"  vers lie
u ";z:z$=z$+"passage("+STR$(x)+","+STR$(
y)+")="+STR$(z)
2850 IF e=7 THEN INPUT"c'est-a-dire:
                       ";entree$:z$=z$+"P
RINT"+CHR$(34)+entree$+CHR$(34)
2860 IF e=8 THEN INPUT"Entrer message po
ur le joueur           ";entree$:z$=z$+"m
$(0)="+CHR$(34)+entree$+CHR$(34)+":GOTO
4500"
2870 IF e=9 THEN z$=z$+"GOTO 4800"
2880 IF e=0 THEN z$=z$+"GOTO 1080":GOSUB
 110:ELSE z$=z$+":":GOTO 2670
2890 INPUT"Encore pour le meme objet ";e
ntree$:entree$=UPPER$(entree$)
2900 IF LEFT$(entree$,1)="O" THEN 2410
2910 IF o<>ao THEN GOTO 2420
2920 o=0:z1=z1+1000:numeroligne=z1
2930 IF v<>av THEN GOTO 2410
2940 CLOSEOUT
2950 CLS
```

```
2960 PRINT"Sur l'enregistreur de donnees
 utilise   il y a maintenant un programm
e portant  le nom":PRINT:PRINT MID$(aven
ture$,2);"  que vous pouvez charger de l
a facon habituelle.":PRINT:PRINT:PRINT:P
RINT"Amusez-vous bien !":PRINT:PRINT:END
```

PROGRAMMEUR
DE GRAPHISME

DESCRIPTION

PROGRAMMEUR DE GRAPHISME

J'ai personnellement développé le programme dont le listing vous est ici proposé afin de pouvoir programmer facilement et rapidement des dessins simples en HAUTE RESOLUTION.

On peut imaginer très simplement la démarche: vous construisez le dessin sur l'écran en utilisant un curseur commandé par des touches et à l'aide de questions posées par l'éditeur. Les données nécessaires à la construction de cette image sont mémorisées et utilisées plus tard pour l'élaboration d'un programme qui peut être ajouté comme sous-programme à d'autres programmes.
Comme cela vous n'êtes pas obligé de déterminer les coordonnées voulues avec des accessoires comme par exemple des feuilles de travail sur écran.

CREATION DE GRAPHISME AVEC GRAFPRO

Après l'en-tête, l'éditeur vous montre d'abord le menu d'aide que vous pouvez atteindre à tout moment en appuyant sur la touche H.
Dès que vous vous êtes informé des possibilités qui s'offrent à vous, vous pouvez, en appuyant sur une touche de votre choix, passer dans le mode d'édition.

Vous disposez maintenant de toute une série d'instructions de caractères et de commande.
Les instructions de caractères sont appelées au moyen des lettres minuscules, les instructions de commande avec les majuscules.

DEPLACEMENT DU CURSEUR :

Le curseur est dans l'écran graphique un point qui ne détruit rien et qui clignote rapidement. Pour le commander on utilise les touches du pavé

numérique:

7 8 9
4 5 6
1 2 3

dont la disposition sur le clavier indique les différentes directions.
Si vous appuyez sur le cinq, le curseur se place au milieu de l'écran.
En début de programme il se trouve toujours sur la position 0,0, à savoir dans le coin inférieur gauche.
Afin que la commande d'un point sur l'écran ne se transforme pas en jeu de patience, vous avez la possibilité de fixer à votre guise l'écart de progression du curseur après avoir entré C.

DESSINER

Pour dessiner un point, amenez le curseur sur la position voulue et appuyez sur P pour Point.
Le point est mis aussitôt, les coordonnées correspondantes sont mémorisées pour la programmation ultérieure.

Si vous souhaitez dessiner une ligne, appuyez sur la touche L et elle apparaît depuis les coordonnées du dernier point mis, jusqu'à la position du curseur.
Pour cela il est indifférent que le dernier point ait été mis avec P ou qu'il s'agisse du point final d'une ligne tracée auparavant.

D reconstruit à tout moment (par exemple après le chargement) une image se trouvant dans la mémoire.

TRAITEMENT DES ERREURS

Si vous avez fait une erreur en dessinant, votre travail n'est pas encore perdu.
En entrant vous effacez l'écran et une nouvelle image apparaît, toutefois sans la ligne dessinée auparavant et sans le point mis en dernier. Vous pouvez répéter ce processus autant de fois que vous le voulez.

SAUVEGARDE ET CHARGEMENT DE L'IMAGE

Appuyez sur L ou S et entrez pour l'image un nom de seize lettres maximum. Grafpro exécute la procédure voulue après que vous ayez appuyé sur ENTER.

PROGRAMMATION DE L'IMAGE

Après avoir appuyé sur P, l'entrée d'un nom est à nouveau nécessaire. On vous demande alors le numéro de ligne par laquelle le programme doit commencer et un programme correspondant est produit.

UTILISATION DES PROGRAMMES CONSTRUITS

Les programmes réalisés ont été placés sur l'enregistreur de données utilisé, sous forme d'un fichier ASCII. Ils peuvent être ajoutés à des programmes existants.
Veillez s'il vous plaît à ce que cependant le processus de chargement soit bien introduit par MERGE. En outre aucune ligne ne doit exister concurremment dans les deux programmes, car sinon les numéros de ligne du programme principal sont sautés pendant le processus de chargement.

Si le programme d'image est chargé avec LOAD, le contenu initial de la mémoire est effacé.

```
1 REM ********************************
2 REM * Grafics Programmer Version 1m *
3 REM * (c) 11.84  by Joerg Walkowiak *
4 REM ********************************
5 CLS
30 DEFINT a-z
40 DIM x(200),y(200),modus$(200)
50 c=10:couleur=1
120 LOCATE 4,5:PRINT"Grafics - Programme
r GRAFPRO CPC 1m"
130 LOCATE 13,6:PRINT"by J";CHR$(178);"r
g Walkowiak"
135 LOCATE 1,12:PRINT STRING$(40,154)
140 LOCATE 10,10:PRINT CHR$(164);" 1984
by DATA BECKER"
150 LOCATE 9,14:PEN 2:PRINT"Tire du livr
e DATA BECKER ":LOCATE 16,16:PRINT"Jeux
d'aventure.":PEN 1
190 FOR i=1 TO 20000:NEXT i:GOTO 2000
1000 entree$=INKEY$
1020 IF entree$="8" THEN Y=Y+C:GOTO 1900
1030 IF entree$="6"THEN X=X+C:GOTO 1900
1040 IF entree$="4"THEN X=X-C:GOTO 1900
1050 IF entree$="2"THEN Y=Y-C:GOTO 1900
1100 IF entree$="p" THEN x(p)=x:y(p)=y:m
odus$(p)="p":PLOT x,y,couleur:p=p+1:lx=x
:ly=y:GOTO 1000
1105 IF entree$="l" THEN  modus$(p)="l":
x(p)=x:y(p)=y:PLOT lx,ly,couleur:DRAW x,
y,couleur:lx=x:ly=y:p=p+1:GOTO 1000

1110 IF entree$="P" THEN GOTO 3010
1120 IF entree$="d" THEN GOTO 6010
1130 IF entree$="S" THEN 4010
1150 IF entree$="L" THEN 5010
1160 IF entree$="9"THEN x=x+c:y=y+c:GOTO
 1900
1170 IF entree$="7"THEN x=x-c:y=y+c:GOTO
 1900
1180 IF entree$="1"THEN x=x-c:y=y-c:GOTO
 1900
1190 IF entree$="3"THEN x=x+c:y=y-c:GOTO
 1900
```

```
1200 IF (entree$="h" OR entree$="H") THE
N GOTO 2000
1210 IF entree$="5"THEN x=320:y=200:GOTO
 1900
1220 IF entree$="c" THEN CLS:INPUT"nouve
au pas du curseur";c:GOTO 6010
1230 IF entree$="@"THEN p=p-1:GOTO 6010
1900 couleur1=TEST(x,y)
1910 FOR x1=1 TO 10
1920 PLOT x,y,couleur1+1
1930 PLOT x,y,couleur1
1940 NEXT x1
1950 GOTO 1000
1999 REM ********************** inst
2000 MODE 1:CLS:LOCATE 15,1:PRINT"  Menu
 HELP":PRINT STRING$(40,154);
2010 PRINT"7  8  9     ";:PEN 2:PRINT"De
placement du curseur":PEN 1
2020 PRINT"4     6     suivant la"
2030 PRINT"1  2  3     disposition des t
ouches                 sur le pave numer
ique"
2040 PRINT"   5  Curseur dans le centre
de l'ecran":PRINT
2050 PEN 2: PRINT"Instructions de dessin
";:PEN 1:PRINT" avec lettres minuscules:
"
2060 PRINT" p - fixer point sur position
 du curseur";
2070 PRINT" l - dessiner ligne a partir
du dernier      point fixe ou de la fin
d'une ligne";
2080 PRINT" d - dessiner nouvelle image"
2090 PRINT" c - fixer pas du curseur"
2100 PRINT
2110 PEN 2:PRINT"Instructions de command
e";:PEN 1:PRINT" avec lettres majuscules
:"
2120 PRINT" H - Menu HELP
2130 PRINT" S - Sauver donnees image
2140 PRINT" L - Charger donnees image
2150 PRINT" P - produire programme Basic
"
```

```
2160 PEN 2:PRINT" @ - rend derniere inst
r. inoperante":PEN 1
2170 PRINT STRING$(40,154);:PRINT"Si  O.
K., appuyez sur une touche.";
2180 IF INKEY$="" THEN 2180
2190 CLS:x=320:y=200:GOTO 1000
3000 REM ********************** prog
3010 CLS:PRINT"programmer le graphisme .
..":PRINT STRING$(40,154)
3020 INPUT"Comment doit s'appeler le pro
gramme";nom$:IF LEN(nom$)>16 THEN GOTO 3
010
3030 IF nom$="@"THEN GOTO 6010
3040 PRINT:INPUT"Numero de la premiere l
igne de programme";ligne
3050 PRINT:INPUT"Quel espacement de lign
es voulez-vous";espacementlignes:PRINT
3060 WINDOW #1,1,40,18,25
3070 OPENOUT nom$
3080 FOR i=0 TO p-1
3090 IF modus$(i)="p"THEN instruction$="
plot":GOTO 3110
3100 IF modus$(i)="l"THEN instruction$="
draw"
3110 ligne$=instruction$+STR$(x(i))+","+
STR$(y(i))
3120 IF LEN(progligne$)<2 THEN progligne
$=ligne$:ligne$="":GOTO 3160
3130 IF (LEN(progligne$)<200 AND LEN(pro
gligne$)>5) THEN progligne$=progligne$+"
:"+ligne$:ligne$="":GOTO 3160
3140 PEN 3:PRINT#9,ligne;progligne$:PEN
2:PRINT#1,ligne;progligne$
3150 ligne=ligne+espacementlignes:progli
gne$=ligne$
3160 NEXT i
3170 IF LEN(progligne$)>1 THEN PEN 3:PRI
NT#9,ligne;progligne$:PEN 2:PRINT#1,lign
e;progligne$
3180 PEN 3:CLOSEOUT
3190 PEN 1
```

```
3200 GOTO 6010
4000 REM ********************** save
4010 CLS:PRINT"Sauvegarde des donnees du
 dessin ...":PRINT STRING$(40,"_")
4020 INPUT"Quel doit etre le nom du dess
in";nom$:IF LEN(nom$)>16 THEN GOTO 4010
4030 IF nom$="@"THEN GOTO 6010
4040 OPENOUT nom$
4050 PEN 3:PRINT
4060 PRINT#9,p
4070 FOR i=0 TO p
4080 PRINT #9,x(i),y(i),modus$(i)
4090 NEXT i
4100 CLOSEOUT
4110 PEN 1
4120 GOTO 6010
5000 REM ********************** load
5010 CLS:PRINT"Chargement des donnees du
 dessin ...":PRINT STRING$(40,"_")
5020 INPUT"Quel dessin";nom$:IF LEN(nom$
)>16 THEN GOTO 5010
5030 IF nom$="@"THEN GOTO 6010
5040 PRINT:PEN 2
5050 OPENIN nom$
5060 INPUT#9,p
5070 FOR i=0 TO p-1
5080 INPUT#9,x(i),y(i),modus$(i)
5090 NEXT i
5100 CLOSEIN
5110 PEN 1
6000 REM ********************** dessin
6010 CLS:FOR i=0 TO p-1
6020 IF modus$(i)="p" THEN PLOT x(i),y(i
),couleur:GOTO 6040
6030 IF modus$(i)="l" THEN DRAW x(i),y(i
),couleur
6040 NEXT i
6050 GOTO 1000
```

A N N E X E

AMELIORATIONS DES PROGRAMMES

Les programmes imprimés dans cet ouvrage doivent vous motiver et vous donner des exemples pour votre propre travail, c'est-à-dire que vous devez pouvoir en avoir une vision d'ensemble afin de pouvoir vous y retrouver sans trop de problèmes.
Mais malheureusement des programmes facilement compréhensibles pour nous, tout au moins quand ils sont en Basic, ne représentent en aucun cas un optimum pour l'ordinateur.
Pour cette raison cela vaudra le coup aussi pour vous de remanier les logiciels présentés ici en tenant compte des aspects suivants.

Il est évident que ce que l'on dit ici vaut non seulement pour nos jeux d'aventure mais aussi pour tous les programmes BASIC !

OPTIMISATION DE LA PLACE EN MEMOIRE

Veillez à remplir au maximum les différentes lignes de programme. Utilisez ainsi le plus petit nombre possible de lignes.
Car chaque ligne d'instruction occupe plusieurs octets pour son propre numéro de ligne, de plus les données doivent être stockées pour retrouver les lignes de programme suivantes.

Renoncez à toutes les REMarques !

A la place des grands noms de variables que nous avons utilisés pour augmenter la lisibilité, utilisez des expressions de une ou deux lettres seulement !

OPTIMISATION DU TEMPS D'EXECUTION

Dans le traitement professionnel des données, pour des raisons de coût et dans notre cas pour un plus grand confort de jeu, on recherche des

programmes qui s'exécutent très rapidement.
Le seul chemin menant à ce but nous contraint à prendre en compte les exigences de l'ordinateur et de son système d'exploitation.

Les propositions suivantes aideront l'interpréteur Basic à augmenter la rapidité de vos programmes. Bien sûr, les propositions précédentes augmentent aussi la vitesse d'exécution.

L'interpréteur construit une série de listes où l'ordre des différents éléments dépend de l'ordre de leur entrée dans la liste correspondante.
Ainsi une place est affectée à chaque variable, dans l'ordre de son apparition, dans la zone de la mémoire programme prévue à cet effet.
Si l'ordinateur doit lire ou écrire une certaine variable, il doit repasser sur tous les élément de cette liste jusqu'à ce que la variable demandée soit atteinte.
C'est pourquoi il est opportun d'introduire les variables suivant leur fréquence, ce qui peut être fait avec une ligne de programme dans laquelle ces variables sont mises par exemple sur zéro.

En outre vous devriez définir auparavant aussi des constantes utilisées fréquemment comme variables car ces nombres fixes doivent être convertis à chaque opération dans le format de traitement de l'interpréteur, ce qui sinon ne se produit que lors de la définition de la variable!

Tenez surtout compte de ce conseil lors de la réalisation de boucles !

Pour le format de traitement il convient en outre de dire que des nombres entiers sont à coup sûr plus rapides à traiter que des nombres à virgule flottante.
Pour cette raison vous devriez commencer chaque programme par DEFINT a-z.
Car de toute façon le nombre des variables nécessitant une plus grande précision sera inférieur, dans pratiquement tous les programmes, au nombre de variables qui n'auront à exprimer que des nombres entiers.

AMSTRAD NOUS VOILA !

LA BIBLE DU PROGRAMMEUR DE L'AMSTRAD CPC

LA BIBLE DE L'AMSTRAD CPC est une aide indispensable pour les programmeurs en BASIC et le MUST absolu pour les programmeurs en assembleur. Cet ouvrage de référence qui révèle vraiment tous les secrets du CPC, est le fruit d'un travail minutieux de plusieurs mois.
Contenu :
- organisation de la mémoire - le processeur - particularités du Z 80, du CPC - GATE ARRAY - le contrôleur vidéo - la ROM vidéo - le CHIP sonore - les interfaces - les systèmes d'exploitation - utilisation des routines avec l'exemple du HARD COPY - le générateur de caractères - l'interpréteur BASIC - BASIC et langage machine - le listing de la ROM - etc.

Prix 249 F TTC Ref ML 122

LE BASIC AU BOUT DES DOIGTS CPC 464

Ce livre est une introduction complète et didactique au BASIC du micro-ordinateur AMSTRAD CPC 464.
Il permet d'apprendre rapidement et facilement la programmation (instructions BASIC, analyses des problèmes, algorythmes complexes...).
Principaux thèmes abordés :
• Les bases de la programmation • Bit, Octet, ASCII • Instructions du BASIC • Organigrammes • Les fenêtres • Programmes BASIC plus poussés • Le programme et menus.
Comprenant de nombreux exemples, ce livre vous assure un apprentissage simple et efficace du BASIC CPC 464.

Prix : 149 F TTC. Réf. : ML 118

AMSTRAD CPC 464. TRUCS ET ASTUCES

De nombreux trucs et astuces pour le CPC 464. La structure hardware, du système d'exploitation, des tokens BASIC, du dessin avec le joystick, des applications de la technique des fenêtres, et d'un grand nombre de programmes intéressants tels qu'une gestion de fichier complète, d'un éditeur de son, d'un générateur de caractères commodes jusqu'aux listings complets de jeux passionnants.

Prix : 149 F TTC Réf. : ML 112

LES JEUX D'AVENTURES. COMMENT LES PROGRAMMER.

Voici la clé du monde de l'aventure. Ce livre fournit un système d'aventures complet, avec éditeur, interpréteur, routines utilitaires et fichiers de jeux. Ainsi qu'un générateur d'aventures pour programmer vous-même facilement vos jeux d'aventures. Avec, bien sûr, des programmes tout prêts à être tapés.

Prix : 129 F TTC Réf. : ML 121

PEEKS ET POKES DU CPC

Les PEEKS, POKES et CALLS sont une introduction aisée à la compréhension du système d'Exploitation et du Langage Machine de l'AMSTRAD CPC. De nombreuses et intéressantes possibilités de programmation et d'applications sont abordées dans ce livre. Quelques extraits : - Configuration "HARDWARE" du CPC - Système d'exploitation et interpréteur. - PEEK et POKE - CALL - Calcul binaire. - Protection de la mémoire.

Prix : 99 F TTC
Réf. : ML 126.

GRAPHISMES ET SONS DU CPC

L'AMSTRAD CPC dispose de capacités graphiques et sonores exceptionnelles. Ce livre en montre l'utilisation à l'aide de nombreux programmes intéressants et utilitaires. Contenu : - base de programmation graphique - éditeur de police de caractères - "sprites", "shapes", et chaînes - représentations multi-couleurs - calcul des coordonnées - rotations, mouvements - représentations graphiques de fonctions en 3D - D.A.O. (dessin assisté par ordinateur) - synthétiseur - minuteur - enveloppes de son, et beaucoup d'autres choses...

Prix : 129 F TTC Réf. : ML 124.

LE LANGAGE MACHINE POUR L'AMSTRAD CPC

"Le langage machine pour l'AMSTRAD CPC" est fait pour tous ceux qui considère que le BASIC n'est plus ni assez puissant ni assez rapide. Des bases de la programmation en langage machine au mode de travail du processeur Z 80 en passant par une description précise de ses instructions ainsi que l'utilisation des routines systèmes.
Tout est expliqué complètement et avec de nombreux exemples.
Le livre contient des programmes complets : un assembleur, un désassembleur et un moniteur. Grâce à ce livre le langage machine n'aura plus de secret pour vous.

Prix 129 F TTC Ref ML 123

AMSTRAD AUTOFORMATION À L'ASSEMBLEUR EN FRANÇAIS

Contient un livre et un logiciel. Le livre : Cet ouvrage introduit le débutant à la programmation du Z80 grâce à la méthode du Dr WATSON qui selon les critiques vaut son pesant d'or ! Aucune connaissance préalable n'est requise et le but du livre est d'assurer au novice un succès total. A la fin du livre les instructions du Z80 sont expliquées en détail. De nombreux exemples illustrent les différentes étapes du cours alors que des exercices (les solutions sont fournies) testent la compréhension.
Le logiciel :
Un assembleur Z80 complet est livré sur cassette et comprend : • Etiquettes Symboliques • Directives d'Assemblage • Chargement/Sauvegarde • Copie écran • INSERT/DELET. L'assembleur permet d'écrire des programmes facilement en langage d'assemblage puis les transforme en code machine (langage machine). Pour vous aider à comprendre les solutions mathématiques utilisées, une démonstration de l'utilisation des nombres binaires et hexadécimaux est fournie. Un programme utilisant les commandes graphiques additionnelles décrites dans le livre est également fourni.
Prix : 198 F TTC Réf. : ML 126

AMSTRAD OUVRE-TOI

Le bon départ avec le CPC 464 ! Ce livre vous apporte les principales informations sur l'utilisation, les possibilités de connexions du CPC 464 et les rudiments nécessaires pour développer vos propres programmes. C'est le livre idéal pour tous ceux qui veulent pénétrer dans l'univers des micro-ordinateurs avec le CPC 464.

Prix : 99 F TTC Réf. : ML 120

PROGRAMMES BASIC POUR LE CPC 464

Alléssentez votre CPC 464.
Ce livre contient de super programmes, notamment un désassembleur, un éditeur graphique, un éditeur de texte... tous les programmes sont prêts à être tapés et abondamment commentés.

* Prix : 129 F TTC Réf. : ML 119

Attention

MICRO APPLICATION lance sa nouvelle gamme entièrement en français de logiciel utilitaire et professionnel.

Demandez notre catalogue gratuit :

NOM : .....

Prénom : .....

Adresse : .....

Code Postal : .....

NUMÉRO 1 - MAI 85 - Prix 15 F. Belgique 120 FB. Suisse 4 FS.

MICRO INFO

LES PIRATES,
CES GÉNIES MÉCONNUS!

LES PEEKS ET LES POKES
DU COMMODORE 64

AVANT-PREMIÈRE :

LE COMMODORE PC

DU NOUVEAU POUR
L'AMSTRAD CPC 464

DOSSIER : TOUT, TOUT, TOUT SUR LE COMMODORE 128

EXCLUSIF ! L'ATARI ST
n'est plus qu'à 800 KM.

CONCOURS AMSTRAD
50 PRIX À GAGNER
Le programme
le plus drôle !

Cher Client,

Nous avons le plaisir de vous annoncer le lancement de notre revue « MICRO INFO ».

Ce magazine comprend des trucs, des astuces, des conseils, des programmes, des concours, des enquêtes et études.

Dans le premier numéro (84 pages) vous trouverez notamment une enquête sur les pirates, un dossier sur le nouveau Commodore 128 et sur l'unité de disquettes AMSTRAD, un poster des Peeks et Pokes du COMMODORE 64...

Pour recevoir le numéro 1 de MICRO INFO retournez-nous, par chèque ou CCP, la somme de 15 FF + 5 FF de port c'est-à-dire un total de 20 FF à

MICRO APPLICATION
13, rue Sainte-Cécile
75009 PARIS

NOM : .....
Prénom : .....
Adresse : .....
Ville : .....
Code Postal : .....

Signature :

.....

□ Chèque □ CCP
à l'ordre de
MICRO APPLICATION

Date

Achevé d'imprimer en octobre 1985
sur les presses de l'imprimerie Laballery et Cie
58500 Clamecy
Dépôt légal : octobre 1985
N° d'imprimeur : 509083

IMPRIMÉ EN FRANCE

Voici la clé du monde de l'aventure. Ce livre fournit un système d'aventures complet, avec éditeur, interpréteur, routines utilitaires et fichiers de jeux. Ainsi qu'un générateur d'aventures pour programmer vous-même facilement vos jeux d'aventures. Avec, bien sûr, des programmes tout prêts à être tapés. Pour tous ceux qui veulent créer leurs propres jeux :

ISBN : 2-86899-018-5
PRIX : 129 FF

Ref. : ML 121

JEUX D'AVENTURES. COMMENT LES PROGRAMMER

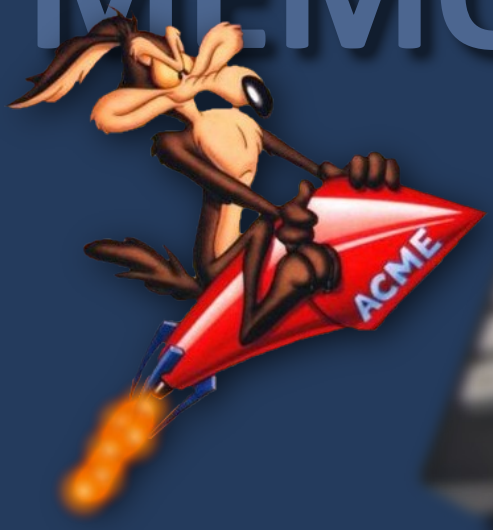
Voici la clé du monde de l'aventure. Ce livre fournit un système d'aventures complet, avec éditeur, interpréteur, routines utilitaires et fichiers de jeux. Ainsi qu'un générateur d'aventures pour programmer vous-mêmes facilement vos jeux d'aventures. Avec, bien sûr, des programmes tout prêts à être tapés.

Réf. : ML 121
Prix : **129** FF

Document numérisé avec amour par :

AMSTRAD CPC

MÉMOIRE ÉCRITE



https://acpc.me/